0-315-03548-X

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Division
Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

51443

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

ROELOF KAMS BROUWER

Date of Birth — Date de naissance

March 15 1942

Country of Birth — Lieu de naissance

The Netherlands

Permanent Address — Résidence fixe

10702 38 Street Edmonton

Title of Thesis — Titre de la thèse

Information Analysis: a Stage in Developing a Data Processing System

University — Université

University of Alberta

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

Master of Business administration

Year this degree conferred — Année d'obtention de ce grade

1981

Name of Supervisor — Nom du directeur de thèse

Dr Peter Winters

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

April 15 1981

Signature

Roelof Brouwer

NL-91 (4/77)

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Ottawa, Canada
K1A 0N4

NL-339 (Rev. 8/80)

THE UNIVERSITY OF ALBERTA

INFORMATION ANALYSIS:
A STAGE IN DEVELOPING
A DATA PROCESSING SYSTEM

by

ROELOF KARS BROUWER

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF BUSINESS ADMINISTRATION AND COMMERCE

FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

EDMONTON, ALBERTA

SPRING, 1981

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Roelof Kars Brouwer

TITLE OF THESIS Information Analysis: A Stage in Developing a Data Processing System

DEGREE FOR WHICH THESIS WAS PRESENTED M.B.A.

YEAR THIS DEGREE GRANTED 1981

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) Roelof K. Brouwer

PERMANENT ADDRESS:

10702 38 Street

Edmonton,

Alberta, Canada

DATED April 15 ....19.81

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Information Analysis: A Stage in Developing a Date Processing System submitted by Roelof Kars Brouwer in partial fulfilment of the requirements for the degree of Master of Business Administration

Peter R. Crowter
.....
Supervisor

.....

.....

Date April 7, 1981

ABSTRACT

The problem in doing analysis of the information and data requirements to be satisfied by any data processing system proposed for development within a business environment is examined. Some effects of not doing proper analysis of information and data requirements are discussed.

First an explanation of how the data processing system or systems fit in the business is provided. Secondly the concept of information as it applies to development of a data processing system is explained. Following this the three most common models proposed by others for doing information and/or data analysis are discussed. This is followed by a proposal for another model called the Entity Role Model. Included also is a suggestion for further research. Finally a complete example of using the Entity Role Model, and the Relational Model to design a hierarchical data base is provided.

ACKNOWLEDGMENTS

Thanks are due to Dr. Peter R. Winters who provided much assistance in the organization of the material and contributed to many ideas.

TABLE OF CONTENTS

CHAPTER		PAGE
1	INTRODUCTION	1
2	THE ROLE OF INFORMATION IN ORGANIZATIONS	8
3	THE ANATOMY OF INFORMATION	16
4	THE ENTITY RELATIONSHIP MODEL	23
5	THE RELATIONSHIP MODEL	32
6	THE MESSAGE MODEL OF LANGEFORS	67
7	A PROPOSAL FOR ANOTHER MODEL: ENTITY ROLE MODEL	72
8	A COMPARATIVE DISCUSSION OF THE VARIOUS MODELS WITH REGARD TO THEIR ROLE IN INFORMATION ANALYSIS	85
9	SUGGESTIONS FOR FURTHER RESEARCH	89
BIBLIOGRAPHY		93
APPENDIX 1 -	AN EXAMPLE OF INFORMATION ANALYSIS BY MESSAGE TYPE DESCRIPTION	101
APPENDIX 2 -	AN EXAMPLE OF CONVERTING MESSAGE TYPE DESCRIPTION INTO HIERARCHICAL DATA BASES	108

CHAPTER 1

INTRODUCTION

The purpose of this report is to consider ways of modelling the data produced by a data processing system. Proper development of a data processing system requires that the system be modelled at its various stages from different points of view. At the present time formal means of describing data processing systems are inadequate. To limit the scope of this thesis, however, only the methods of modelling the information to be produced by a data processing system will be considered. Thus methods of modelling data processing sytems in their entirety will not be considered here.

It has long ago been recognized that even though hardware costs have decreased over the years, the cost of software has not shown the same pattern. Cost overruns for software development projects are frequent. Estimated completion dates for software development projects are set and reset. Even when the software product is completed the user is often left very disappointed with the result. He is left to feel that his requirements as he thought he had identified them have not been met. The user does have available a useful tool for describing the expected output of the system to be built.

Forrester in his book called Industrial Dynamics says

> "Management is in transistion from an art, based only on experience, to a profession, based on an underlying structure of principles and science. Any worthwhile human endeavor emerges first as an art. We succeed before we understand why. The practice of medicine or of engineering began as an empirical art representing only the exercise of judgement based on experience. The development of the underlying sciences was motivated by the need to understand better the foundation on which the art rested. The art develops

> through empirical experience but in time ceases to grow because of the disorganized state of its knowledge. The rapid strides of professional progress come when the structure and principles that integrate individual experiences can be identified and taught explicitly rather than by indirection and diffusion" (FOR-61)

The preceding applies as much to systems analysis work as it does to management. It has often been said that the methodology of system analysis as applied in developing data processing systems is at the stage where engineering was in 1937. It is ironic that data processing systems should be developed in anything but a systematic manner.

Developing a system without systematic approaches works to a limited degree as long as the system to be developed is small enough so as to be perceivable and requires only a few people for development. If the system is so small as to be developed by one person little communication is required during development except with the future user of the system. Formal systems for development tend to supersede informal ones, however, when many people have to cooperate in a complex system, development of which requires precise work over extended time periods. In case the system to be developed is small it need only be described in very general terms during the proposal stage and finally it can be described in terms of the manual and mechanized processes, printed reports and so on. The developer can start with a very general model of the system, making up the proposal and can then turn this model directly into a working or operational system. The developer can proceed directly from the proposal to the operational system because of its size which means the description of the operational system in terms of programs and procedures can be carried in his head.

Let us now think of the project for developing an information system as being an information system in its own right. The project consists of

people carrying out manual procedures which produce information and have information as input. The final primary output of the project consists of the manual and mechanized procedures making up the operating system, the mechanized procedures being the computer programs. The initial primary input of the project system consists of the objectives of the system to be developed. The programs developed are written according to relatively well defined rules and therefore the final output of the project system is relatively well defined. For large, imperceivable systems it is not realistic, however, to directly convert the objectives into programs because the number of things to be considered is excessive and it is not possible to comprehend the system in its totality. The project system should generate intermediate outputs. The project work can be viewed as consisting of stages carried out in sequence with each stage taking the information produced by the preceding stage as its input and producing information as output to be used as input for the succeeding stage. The idea that the development of an information system should consist of stages is not new. The stages are often called feasibility, definition, design, construction and implementation stage. Another sequence of stages is feasibility, preliminary design, detail design, implementation and conversion stage. Attempts have been made to define the activities making up each of the stages but it is only in the last few years that attempts have been made to define the output, also called deliverables, that should be created by each of the stages. Thus although the final product to be produced by the project is well described the products to be produced by the intermediate stages often are not.

We may think of the development of an information system as proceeding from the general to the particular. Accepting the fact that to describe a system is to present a model of it, the development of an information system starts with a very general model, not including much detail, and finishes with a very detailed model which is the system itself. The first model includes only the objectives or functions of the system to be developed. The final model of the system will include devices to carry out procedures, data sets which are inputs and outputs of the procedures and the procedures themselves. Development of the information system begins with a very general explanation of what is supposed to happen and ends with a set of instructions to both machines and people.

Just as it is appropriate to design the product of a machine before designing the machine itself it is also appropriate to design the output of a data processing system before writing the procedures themselves. In fact the inputs should also probably be designed before everything but the outputs. One model of the target system (system to be developed) may consist of information sets and their precedence relationships which indicate which information sets are used to generate another information set. The processes or procedures are implicit rather than explicit in this model.

The purpose of this report is to indicate ways in which the information used by, stored by or produced by the target system may be described. Description of the information should also proceed from the general to the particular. What is needed then is a tool, simple enough so that the user of the target sytem may be able to use it, that may be used to specify the information to be produced by the target system.

This tool should make it possible to define the needed information as well as the information structure independently of any implementation decisions. The tool should be such that it is simple enough to be understood by people who are non-technical, yet precise enough so that the documentation produced may be used unambiguously in other stages of development of the target system. Practices are often such that design is made on a detailed level from the beginning. This means that data and computer programs are designed together with the data being designed to suit a certain program. The problem with going to the detailed level immediately is that in case of a large system there are too many pieces to consider all at once.

If data is the representation of information then this report focusses not so much on how the information is represented in the mechanized data processing system but rather focuses on how the information to be used by, stored by and produced by the target system may be documented. The information analysis document will then serve as input to the design activity.

G.M. Nijssen, in a paper called "Current Issues in Conceptual Schema Concepts" (NIJ-77), defines the Conceptual Schema to be that which "describes which information is considered to be within a specific universe of discourse" while the External Schema "describes how a subset of this informaton is presented to a program" and the Internal Schema "describes how the information is physically represented in storage". The Conceptual Schema is "a set of rules describing which information (or knowledge) may enter and reside in the information base", the Internal Schema is "a set of rules describing how the information (or knowledge) is physically represented on storage media" and External Schema is "a set

of rules describing how information (or knowledge) is viewed by a program".

Sundgren in a paper titled "Conceptual Foundation of the Infological Approach to Data Bases" (SUN-74) discusses what he means by an infological theory of data bases which he feels should "make a clear distinction between (a) the object system (b) the information about the object system (c) the data representations of information and (d) the storing and accessing medium to which the data are allocated". The object system is the system about which the data processing system provides information. An example of an object system is the subsystem for placing telephone poles and cables in a telephone company. The system parts are people, equipment and material. The type of information about the object system could consist of information about the location of the telephone equipment in existence, information about the equipment to be retired or information about the equipment to be placed.

Information however is only a abstraction and still needs to be represented. That which represents information is called data. In this case an example of data is

JOB #	MATERIAL TYPE	QUANTITY
1	A	2
2	B	3

This data may, however, be stored on disks, magnetic tapes or paper. Sundgren considers mappings from a to b, b to c, and from c to d. By splitting the one mapping into three types of mappings different people with different skills or knowledge can be used to define the various mappings in particular instances. Thus the mapping from the object system to the model of the information about the object system could be

defined by the system developer who is very knowledgeable about the object system and also understands what information should be provided about the object system. He needs to know the applications area quite well and understand the method for describing both the object system and the information to be provided. He should not need to know anything about mechanization. This paper will discuss ways of defining the information to be provided by the target system (the data processing system to be developed).

Another author belonging to what may be called the Scandanavian school of thought is Bubenko. He considers three design and description levels (BUB-76). 1. the information structure (or "infological") level, 2. the data structure ("datalogical") level and 3. the storage structure level. In the first level information is referred to "in problem - oriented and implementation independent terms". In the next level the data representation of information is described. Finally the third level is concerned with how and where the data is stored. The mappings are concerned with mapping "reality" into an information structure, the information structure into a data structure and the data structure into a storage structure.

The main thrust of this paper is to describe various Conceptual Schemas with examples. The main schemas to be discussed are the Relational, the one developed by the Scandanavian School, the Entity-Relationship model, and one suggested by the author.

CHAPTER 2

ROLE OF INFORMATION SYSTEMS IN ORGANIZATIONS

Since the thesis is about information systems in business an attempt will be made to define the roles that information systems play in organizations. Understanding these roles is important to understanding information analysis since part of information analysis consists of describing what the information to be provided by the target systems is to be used for. Classification of information systems into different types will serve to demonstrate the various roles information systems play in organizations.

Organizations according to Jay Galbraith (GAL-77) are "(1) composed of people and groups of people (2) in order to achieve some shared purpose (3) through a division of labour (4) integrated by information based decision processes (5) continuously through time". In organizations the overall task of the organization is broken into sub-tasks which can be performed by individuals and groups of individuals. These sub-tasks in turn must be re-integrated into the completion of the whole task. The work may be divided so that the portions are differentiated rather than similar, and so that each individual has a small portion of the overall task. This is often called the horizontal division of labour. This division of labour may be more productive than another division because it overcomes physical limitations and the cognitive knowledge limitation of people. Each person can specialize and requires a smaller scope of knowledge. This division of labour however increases the inter-dependence among sub tasks. The increased inter-dependence in turn creates problems of

coordination. Inter-dependence of tasks increases the amount of information that has to be transferred between the various people carrying out the tasks.

Much of what goes on in organizations is decision making and information processing. The greater the uncertainty of the task, the greater the amount of information that has to be processed between decision makers during the execution of the task. If the task is well understood prior to its performance, much of the activity can be pre-planned. If it is not understood, then during the actual task execution more knowledge is acquired which leads to changes in resource allocation, schedules, and priorities. The basic effect of uncertainty is to limit the ability of the organization to pre-plan or to make decisions about activities in advance of their execution.

Galbraith defines uncertainty to be the difference between the amount of information required to perform the task and the amount of information already possessed by the organization. The amount of information required to perform a task is a function of the nature of the task itself and the level of performance. Higher performance levels necessitates considering more alternatives, more variables, and more variables simultaneously. Different organizing modes are adopted by organizations to deal with task uncertainty due to limited capacities to process information. Variations in organizing modes are, according to Galbraith, actually variations in the capacity of organizations to process information and make decisions about events which cannot be anticipated in advance. In order to coordinate inter-dependent roles, organizations have invented mechanisms for collecting information, deciding, and disseminating information to resolve conflicts and guide inter-dependent

actions. As the task uncertainty increases, the volume of information from the points of action to points of decision making overload the hierarchy.

To solve the problems of information flow either the need for information can be decreased or the capacity for information flow may be increased. The capacity of the decision maker can be increased by employing computers. Variables, according to Galbraith, are decision frequency or time of information flows to and from the decision mechanism, scope of the data base available to the decision mechanism, degree of formalization of the information flow to and from the decision mechanism and capacity of the decision mechanism to process information and select the appropriate alternative. The information flow may be either periodic or continuous. The scope of the data base may either be local or global. If the decision mechanism has access to information pertaining only to its immediate location, the data base is called local. If the decision mechanism has access to information concerning the state of affairs in all resource groups and for all output categories, the data base is global.

The scope of the data base available to the decision mechanism affects the ability to direct activities in one part of the organization so as to be consistent with the activities taking place in other parts of the organization. The greater the inter-dependence between sub units the greater the need for a global data base. The primary effect of the formalized languages is to permit the transmission of information with fewer symbols thereby expanding the communication channels to carry more information. Information systems are classified by Galbraith as local - periodic, local - real time, global - periodic or global - continuous.

The local-periodic information system ignores inter-dependence and is not responsive to uncertain environments. For example the warehouse clerk who places orders every Friday to replenish inventory makes a decision periodically, based upon local information. With a local real time information system decisions are made whenever decisions need to be made. The decision is made in light of the most current information concerning the status of the department. Batch processing computer systems are typical of global - periodic information systems which are used extensively in applications where uncertainty is low to moderate. The on-line real time computer system is typical of the global continuous information system. Examples are airline reservation systems and tickets on systems for sports and theater events.

As an aid to understanding what a management information system entails it maybe useful to classify them into types. Since management information systems are there to support decision making it is possible to classify them according to decision types supported.

A framework developed by Robert N. Anthony (ANT-65) does just that. He considers three distinct types of decision activities in an organization. These three types of decision activities require three distinct supporting information systems. The decision activities are operational control, management control and strategic planning (ANT-65). Strategic planning includes deciding on organizational objectives and on the means for achieving them. The strategic planner is mainly concerned with the relationship between the organization and its environment. Management control includes assuring that resources are used effectively and efficiently in pursuing the objectives specified through strategic planning. Operational control includes assuring that specific tasks are

carried out effectively and efficiently.

A frame-work proposed by Simon (SIM-60) is also based on the decision types, i.e. programmed and non-programmed decisions. Programmed decisions are routine and repetitive. A rule can be specified before hand and the outcome of the decision making process will always be the same for the same input. These decisions are also called reflex. Non-programmed decisions are those for which no exact rule can be specified. These decisions are novel, unstructured and consequential.

Another frame-work for managerial activities and therefore-management information systems is that developed by Gorry and Scott Morton (GOR-60). This framework is really a combination of Anthony and Simon's frameworks. Decisions are classified in two dimensions: one dimension is the structured versus non-structural scale, while the other dimension is the operational control to strategic planning scale. Structured in this case is another word for programmed and unstructured is another word for non-programmed. An information system dealing with accounts receivable would fall in the area of structured operational control since structured decisions are made and specific operations are supported. Personnel management would fall under unstructured-managerial control since assignment of personnel involves a good deal of judgement.

Other frameworks are considered by Blumenthal (BLU-69), Forrester (FOR-61) and Dearden (DEA-65). Other terminology for structured versus non-structured or programmed versus non-programmed decision making is reflex versus discretionary decisions which is used by Grindley in his book on systematics (GRI-75). Decisions taken in the reflex area depend upon rules, which state what instructions to give to the behaviour function under certain conditions. This means that the outcome of a

decision is predefined once the input conditions are known. The decision making can then be automated if the rule consists of making certain data element value comparisons.

Discretionary decisions, on the other hand, are those in which judgement is involved. Because discretionary decisions require some degree of judgement, they will not be consistent from time to time and from person to person. If we consider decision making as an information process then reflex decision making is a process for which inputs, outputs and the fixed relationship between the inputs and outputs can be defined. The process is deterministic in that the same input will always generate the same output. A discretionary decision on the other hand is one where usually only the possible outputs are predefined but neither the input nor the relationship between the outputs and inputs is defined. The process is therefore non-deterministic.

From the described characteristics of the two types of decisions it is immediately obvious that reflex decision making can be automated while discretionary decision making cannot be automated. A mechanized or automated information system can include reflex decision making processes but cannot include discretionary decision making processes. However a mechanized information system can aid the discretionary decision making by providing information as input.

An interesting point is the evolutionary nature of the line separating reflex decisions from discretionary decisions. As we improve our understanding of a particular decision we can move from the discretionary area to the reflex area and allow the automated information system to take care of it, freeing the manager for other tasks. For example some time ago the inventory re-ordering decision in most

organizatons was made by a well-paid member of middle management. Today this decision has moved from the discretionary decision making area to the reflex. The decision rules which in this case are the EOQ (Economic Order Quantity) formulae provide a well defined relationship between input and output of the decision making process.

Another framework to which reference is made often, is one proposed by Langefors (LAN-73). This framework is similar to the one proposed by Anthony. Langefors classifies the different functions of an information system into two kinds according to the tasks supported. For one task type information is necessary, for the other type information is more or less useful, while not being absolutely necessary. In order to determine whether or not to bring data in and use it in the system, one has to estimate both the utility and the data processing cost that would be the result of its introduction. The kind of task for which information is necessary has to do with the operative function of the managed system. Operative information is necessary information. Each operation needs new operative information. If to the requirement of operative functioning a requirement of total system economy is added, then new kinds of information are called for.

The second category is called directive information by Langefors. Input, required to produce output consisting of directive information consists of information from all over the managed system and even from its environment. We note that the framework of Langefors is similar to that of Anthony if we consider that operative information is for operational control and directive information is for management control and strategic planning.

This should have given some idea of the role that data processing systems, which may be manual or mechanized, play in a company. Part of information analysis is to determine what operations are carried out in a company and note what information is required to support these operations in the case of an operational control system.

The next chapter will focus on information in a more microscopic sense. An understanding of information is required so that we may determine how information is to be represented.

CHAPTER 3

THE ANATOMY OF INFORMATION

Since the objective of this report is to discuss various ways of representing the information produced by a data processing system let us first spend some time discussing the meaning of information. How is information different from knowledge and data? What are the different aspects of information? Understanding the difference in meaning of information and data will help in understanding the differences in the various models and also serve to explain information analysis.

We can think of information as consisting of facts about things which may be called entities. Something may be said about a thing by itself or about the thing in relation to another thing. Information is being considered rather than data if the representation is not of interest. Information can be represented by data in various ways. Even though many representations of the same information may exist it is impossible to imagine information without some type of representation. Thus the information that two objects are two inches apart can be represented in the English language or by a picture but it is also represented by the actual objects themselves in relationship to each other. The objects can be represented by themselves or by things which bear no resemblance to the objects at all. The only requirements for representing objects is that if two objects are to be distinguished in some context the representations must be distinct. The representation of some object can vary from closely resembling the object to bearing no physical resemblance to the object at all.

Since we can only perceive things through one of the five senses, information communicated to us must have a physical aspect. A thing, if it can be detected through the visual senses (i.e. if it reflects light), can either be pointed to directly if it is in view of both the sender of the information and the receiver of the information, in which case the object is used to represent itself, or it can be represented by something else which has properties detectable through the physical senses. The most commonly used senses are sight and hearing and patterns on paper are often used to represent information. Another representation consists of magnetized spots on a magnetic tape. It is interesting to note that although the thing represented does not have to have physical aspects its representation must. Properties of physical things, for example, are in themselves not physical but their representations are.

Even though the fact that two objects are 2" apart can be represented by the objects themselves in relation to each other the fact that the two objects were 2" apart at some previous time requires a different presentation. It may be said that it is only when the state described is the present state that the thing itself can represent the information about itself. When information describes state change, past state, or future state it is difficult to think of its existence without some form of external representation. An account of an event, ie. state change, might be given orally or in writing in some natural language by means of free informal sketches or by more or less formalized methods, in some cases even including mathematical formulas. One of the things which cannot be dealt with without some form of representation is the passage of time. State change cannot occur without the passage of time and the passage of time cannot occur without change in state. In fact when we

say that a change in state took place during a certain passage of time we are really comparing the state change of one thing to the state change of something used as a standard (say the moon as observed from a certain location on earth).

Information may be studied from various points of view. The three main aspects which often are considered are the syntactic, the semantic and the pragmatic. Syntactics is the study of signs and the relation between signs. It is only concerned with the physical signs abstracted from the user. Semantics is concerned with the study of the relationship between signs and their designata, independent of the specific communication effects. Pragmatics on the other hand is the study of signs in relation to their users. It is concerned with the effect that the information has on the receiver. In this report the pragmatic aspects will not be discussed.

An interesting discussion on the relationship among knowledge, information and data is provided by Lindgreen (LIN-74). He refers to the word knowledge as meaning "what has been recorded somehow in the human brain". Four fundamental informatic processes are considered:

knowledge --- formalize --- information
information --- represent --- data
data --- recognize --- information
information --- evaluate --- knowledge

Communication is considered to involve two of the above processes which are representing and recognizing ie.

information -- represent -- data -- recognize -- information

One of the ways of further analyzing information is to determine the minimum requirements for a collection of symbols to represent

information. Langefors (LAN-69) uses the term elementary message for the smallest unit of information. A further explanation of what is the smallest unit of information will be provided when the model of Langefors is discussed in Chapter 6. An elementary message cannot be decomposed further and still provide information. Let us now consider various types of messages.

Information generally is about systems consisting of entities related to each other in various ways. Information may therefore provide the property value of some object which holds at the present time, held in the past or is expected to hold in the future. Information may also define the relationship which held between two objects in the past, holds between two objects at the present time or is expected to hold between two objects at some future date. Information may include the identification of an entity together with a value for a property type corresponding to the entity identified. On the other hand only the existence of an entity with a particular value for property may be indicated. Thus the statement "there are 4 items of type widget" indicates that there exists a set of objects such that there are 4 items in the set and the objects in the set all have the property of being of type widget (ie. the objects have the properties shared with widgets).

Objects belonging to the universe of discourse (ie. the set of objects to be informed about by the information system) may be identified or labelled in several different ways. One way would be to consider all the entities together as making up one set and then assign a similar but distinct label to each entity. Depending upon the number of entities to be labelled a 1 digit, 2 digit or n digit number could be used for each label. A more common approach is to classify the entities into entity

types. Only the entities belonging to one entity type need to be distinguished since identification of an entity is two fold consisting of identification of entity type and further identification within the entity type. For example the employee with employee number equal to 2000. The word employee identifies a set of entities while the number 2000 identifies a particular entity within this set. Often more than one scheme for labelling the entities within an entity set is used which implies that identification of an entity consists of three parts including (1) specifying the entity set (2) specifying the labelling scheme used and (3) specifying a value for the label variable. Thus "employee number" and "employee name" deal with the same entity sets but with different labelling schemes. The reason for having the two labelling schemes is that the information system designer has no control over the labelling scheme implied by employee name and thus there may not be the desirable one to one correspondence between label and entity. On the other hand the first labelling scheme will be better known and has a wider area of use.

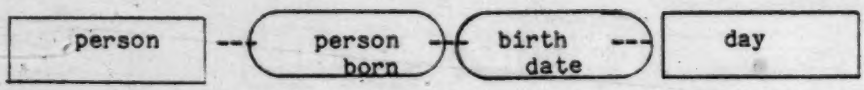
Often it is not required to distinguish between entities when to all intents and purposes the entities are identical. If we imagine the set of all these entities which are identical the material type identifier can be considered as the identifier of this set. Thus the statement "there are 5 items of type "a" at location 6 at the present time" can be rephrased as "There is a set of objects at location 6. The properties of this set are (1) there are 5 elements in the set (2) each object in the set has the properties shared by those elements belonging to set a". We note that in this case an object is not uniquely identified. The statement is about 2 entities and their relationship. One of the

entities is a location which is uniquely identified and the other entity is a set which is not uniquely identified but whose properties are given.

Let us now consider types of elementary information kinds or messages. First of all a message may describe the appearance of something at some particular point in time, which may be past, present or future. Thus a message may describe state. Secondly a message may describe the change in something which took place between two points in time. This may be called state change. For example to say that an object is at a particular location is to describe state, to say that a particular object has been moved from one location to another is to describe state change. An elementary message may a) identify an object and give a property for that object which holds at a particular time b) indicate that an object with specific properties exists at a particular point in time c) indicate that a relationship holds between two or more objects. Time must be included as part of every message either explicitly or implicitly. This is because things in general don't remain constant.

Another but similar approach to defining the basic component of information is described by Nijssen (NIJ-77). In his paper he defines what is called a semantically irreducable sentence, an atomic sentence or an elementary sentence. All three terms stand for the same thing which is similar to the elementary message defined in Langefor's paper. According to Nijssen "information is a set of one or more sentences. A sentence is a set of one or more related elementary sentences. An elementary sentence is a set of one or more object role pairs, and each role appears at most once in the set." (NIJ-77) Nijssen further defines an atomic object as a entity which is not further considered to be

divisable. The role is the function that atomic object plays in the atomic sentence. For example the sentence "John Bergsma's birthdate is Feb 2nd 1940" has two atomic objects in it. The one is identified by the name John Bergsma and the other is identified by the name Feb 2nd 1940. The first atomic object is of type person while the second atomic object is of type day. The first atomic object plays the role of person born while the second atomic object plays the role of day born on. A more precise definition of atomic sentence provided by Nijssen (NIJ-77) is "an atomic sentence is a set of one or more pairs, where each pair consists of an atomic object reference and a role reference, and each role appears at most once in the atomic sentence". The notation used by Nijssen to denote or describe atomic sentences is as shown below. The ellipse denotes an atomic object type while the rectangle denotes role. Consider for example:



The diagram above in actuality depicts an atomic sentence type rather than a instance of an atomic sentence type. Two atomic sentences belong to the same atomic sentence if both include the same types of atomic objects and if the atomic objects belonging to the same types in the two sentences also play the same role.

Having examined the difference between information and data we may now start examining the different methodologies for information analysis or data analysis.

CHAPTER 4

THE ENTITY RELATIONSHIP MODEL

The entity - relationship approach to information modelling as described by Dr P. Chen in two of his papers (CHE-76 and CHE-77) will now be considered. The entity relationship model considers both entities and relationships between entities as the main objects. The model comes out of a four level view of data. The four levels according to Dr. Chen are

1) information concerning entities and relationships which exist in our minds.
2) information structure - the organization of information in which entities and relationships are represented by data.
3) access-path-independent data structure - the data structures which are not involved with search schemes, indexing schemes etc.
4) access-path dependent data structure.

The entities and relationships at level 1 are considered to be conceptual objects in our minds while at level 2 representations of the conceptual objects are included. Since it is impossible to talk about objects without some form of representation it is also difficult to work at level one. However what must be remembered is that even though at level one we represent the entities and relationships by data, the data used for the representation is not the main item of interest.

In the entity relationship model information is considered to be information about entities and about entity relationships. Thus for example an entity could be an employee while the information about this entity is the employee's employee number, his age, his home address and so on. The information about entities can be divided into property types

or attributes with corresponding attribute values. Thus "employee number is equal to 3000" says that the property "employee number" has the value 3000. To organize the information about entities, entities are divided into entity types. All the entities within an entity type set can be described by the same set of property types. Thus the entities making up the set of employees all have similar property types such as age, birth date, salary and so on.

A second object type considered by Chen to be part of his model is the relationship between two or more entities. An example is the relationship between an employee and a project where the employee is manager of the project. Another example is the relationship between a project and an employee where the employee is a worker on the project.

Two relationship types are considered to be distinct if either the component entity types are different or if the roles played by the entity types are different. Thus the two relationship types mentioned above are distinct. Information may also consist also of property values for a relationship. A relationship is identified by identifying both the entities involved in the relationship and also by identifying the roles played by the entities. Chen, in his model, considers the identifier of an entity to be a property of the entity as much as say its colour. Thus "employee number equal to 3000" is a property of an employee. Similarly for relationships. Chen uses the word relationship both for the association of entities and also for the association of the property values. Let us now assume that unless stated otherwise the latter meaning is intended.

Chen considers two forms of entity relations where an entity relation is a set of relationships in the mathematical sense such that each

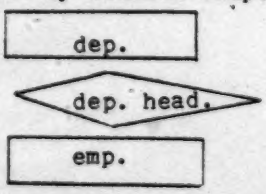
element of the relation provides the properties of one entity. If relationships are used for identifying the entities the relation is called a weak entity relation. If relationships are not used for identifying the entities the relation is called a regular relation. As an example consider the relation consisting of relationships which are associations of property values for streets. A street is identified by country, city and street name and therefore the relation consisting of these relationships would be a weak entity relation.

Chen also considers two forms of entity relationship relations where an entity relationship relation is a set of relationships such that each element provides the properties of a relationship of entities. In his own words: "If all entities in the relationship are identified by their own attribute values it is called a regular relationship relation; while if some entities in the relationship are identified by other relationships the relation is called a weak relationship relation" (CHE-76). An example of a weak relationship relation is one which provides the properties of associations between departments and divisions within the departments. In this relation the divisions are identified by both the division name and the department name.

In his paper titled "the entitiy relationship approach to logical data base design" Chen (CHE-77) considers enterprise schema, external schema and internal schema as models of data. The enterprise schema is the enterprise view of the data, the external schema is program oriented while the internal schema deals with the physical storage of data. Of the three the enterprise schema should be least implementation oriented. In Chen's words: "The enterprise schema should be pure representation of the real world and should be independent of storage and efficiency

considerations" (CHE-77). Chen proposes an entity - relationship (e-r) diagrammatic technique for identifying the entities and relationships (associations of entities) which are of interest to the enterprise. The advantages of developing the enterprise schema before logical or physical data base design is carried out is that the enterprise schema is more user oriented, is not restricted by capabilities of the data base system and is more stable than the user schema. The concept of the enterprise schema is similar to the concept of conceptual schema as proposed by the Committee on Computer and Information Processing of the American National Standards Institute (ANSI). This committee defines the conceptual schema to be the enterprise view of data. In their words: "It is a description of a model of the enterprise in terms of its entities, attributes, and the relationships among them" (CHE-77). Corresponding to the different views of data the aforementioned committee also identified three types of data base administrators which are the enterprise administrator, the data base administrator and the application administrator. The enterprise administrator is responsible for modeling the enterprise, the data base administrator for the physical data base and the application administrator is responsible for application programs.

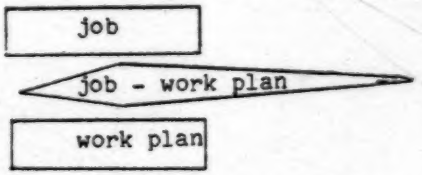
Chen's diagrammatic technique of describing the enterprise view consists of representing entity types by rectangular boxes and entity relationship types by diamond shaped boxes. An example is as shown below.



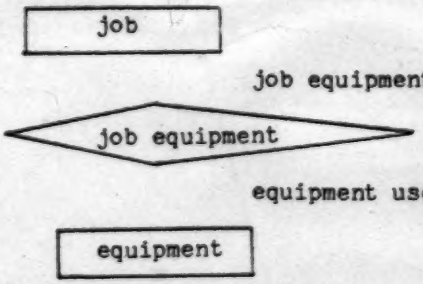
The entity types are "department" and "employee" while the association type is "department head". A property of the association

type is the number of different occurrences of the one entity type which are associated with one occurrence of the other entity type. Thus the association in the example is one to one since with one occurrence of department there is associated with it one occurrence of employee in the department head association and with one occurrence of employee there is associated with it one occurrence of department.

An example of a one to many relationship between entity types is:

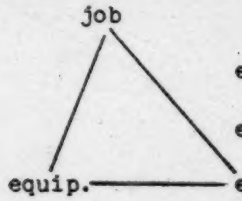


The entity type "job" corresponds to those entities which consist of a set of activities and "work plan" corresponds to subsets of those activities. Associated with each job there may be one or more work plans but associated with each work plan there is associated exactly one job. Since a work plan cannot exist without a job the work plan is existence dependent upon job. If the work plan is only uniquely identified in the set of work plans for one job but not uniquely identified in the set of all work plans without specifying the job, the work plan entity type is considered to be identification dependent upon the job entity type. An example of a many to many relationship is the one following.



One job can use many pieces of equipment and one piece of equipment can be used on many jobs.

Up to now only relationships consisting of two entity types have been illustrated. An example of a relationship consisting of three entity types is the following.



equipment operated by employee on job.

The ternary association can be decomposed into several binary associations for the purpose of defining the many to many property. Let the letter "j" stand for job, "m" for employee and "e" for equipment. The binary association types are

j<->e

j<->m

e<->m

j-e<->m

j-m<->e

e-m<->j

Where e-m is the association between employees and pieces of equipment treated as an object type. If we assume that many types of equipment can be used on one job, one type of equipment can be used on many jobs, several employees can work on one job, one employee can work on several jobs, one type of equipment can have many operators and one employee can operate many types of equipment the properties of the association are

```
M     N
j<->e

M     N
j<->m

M     N
e<->m
```

The preceding also has the following implications.

```
M     N                    M
j<->e  implies  j -> e-m

                           M
          implies  e -> j-m

M     N                    M
j<->m  implies  j -> e-m

                           M
          implies  m -> j-e

M     N                    M
e<->m  implies  e -> j-m

                           M
          implies  m -> j-e
```

M
Where j -> e-m means that for one job there can be many equipment operator combinations.

The significance of the type of mapping (ie whether one-to-one, one-to-many, or many-to-many) lies in the data structure used to store the information as will be seen later.

After the portion of the enterprise (object system) to be monitored by the data processing system (target system) has been described in terms of entity types and entity association types the next stage of refinement involves defining the information types about these entity types and association types which are of interest for monitoring the object system. The type of information about an entity type can be defined in

terms of properties and values for these properties. For example if the entity type is job then properties of the job are "date - construction start", "date - construction complete", "date - job closure", "job cost - budgeted", "job cost - actual" and so on. Each of these properties has a value depending upon the particular entity.

If jobs are assigned numbers for identification purposes, the "job number" is also regarded as a property by Chen. This writer however feels that the thing used to represent an entity, like "job number" does a job, is not really a property in the sense of the other properties but instead plays the role of surrogate. It serves as a sort of anchor point to which other data can be attached. The property values provide information about a certain entity. This entity is either an abstraction or otherwise cannot be pointed to directly and therefore some name has to be used to serve as a link between the entity itself and the property value representations about the entity.

Values may be classified into different value types such as dates, costs and so on. In the example above the values for the property called "date - construction start" come from the same set as the values for the property called "date - construction complete". The two properties called "date - construction start" and "date - construction complete" may be examined from another point of view. If we consider a "day in time" to be an entity then a value for "date - construction start" identifies one of these entities and "date - construction complete" identifies an occurrence from the same set of entities. Thus we can think of an association between a job and a "day in time" where the "day in time" plays the role of construction start and think of another association between a job and a "day in time" where the "day in time" plays the role

of construction complete. This illustrates why although there are two properties there is only one property value set. This example also illustrates that although "date - construction start" was considered to be a property of job we can also think of "job number - date" as representing an association between a job and a "day in time". The reason that the property point of view is more acceptable is because the entity day is of no interest in its own right.

Up to this point only attributes of entities have been considered. The properties of associations are also of interest however. Consider for example the two entity types job and account. The account is an entity used to classify cost associated with an enterprise. The association between a job and an account signifies that certain costs associated with a job fall in a certain class. A property of the association between job and account is the number of dollars representing the partial cost of the job associated with the account. An example of a property of a ternary relationship is the cost associated with a job, account, and material type combination. The property is the cost associated with the job, account, material type combination.

Although it is more common for the association between entity type and property type to be many to one, it is also possible to have many to many associations thus for example a type of material may be available in several colours. It is possible however to change this into a many to one situation by considering each of the colours associated with a particular material type to be the value of a different property or the same properties playing different roles. The properties could become most common colour for the item, second most common colour and so on.

CHAPTER 5

THE RELATIONSHIP MODEL

Introduction

Another method for describing information sets is the Relationship Model proposed by Codd (COD-70). An information set may be all the information represented by data in a data base or the information represented by data on forms. The method has as its basis the mathematical entity called a relation.

Meaning of relation

In mathematics a relation is a set of tuples all having the same number of elements such that the elements which are in the first position are from one set, the elements in the second position are from another set and so on. If S_1, S_2 -- S_n is a family of sets then S_1 x S_2 x --S_n denotes the set of all possible tuples such that the first component is an element of S_1, the second component is an element of S_2 and so on. The set of all possible tuples above is called the Cartesian product of S_1, S_2 -- S_n. Every relation is a subset of some Cartesian product. It is useful to think of a relation as a set of associations between values of a set of variables. Thus each component of a particular tuple can be considered as the value of a variable. For a particular relation all the first components are the values of one variable, all the second components are the values of another variable and so on. In this way the relation defines associations between values of variables. As an example consider the three variables whose values denote position in an x-y-z coordinate system. The variables can be

denoted by x, y and z. The relation may consist of the tuples (1,2,3), (1,1,4) and (6,3,2) which corresponds to three positions in space. The first component in each tuple corresponds to values of the variable "x", the second component to the variable "y" while the third component corresponds to values for the variable "z". It should be noted that the same information can be expressed by [(x = 1, y = 2, z = 3), (z = 4, y = 1, x = 1), (y = 3, x = 6, y = 2)]. Strictly speaking the preceding set is not considered to be a mathematical relation. The reason for the first form as opposed to the second form is of course brevity. In the first case since position within the tuple corresponds to a particular variable of which the component is a value the variable does not have to be stated explicitly. Let us now consider the relation [(1,3,2), (1,4,1), (6,2,3)] with the first component corresponding to values of x, the second component corresponding to values of z, and the third component corresonding to values of y. Although the third and the first relation express the same information, the relations are not considered to be the same from a mathematical point of view .

In the discussion which follows all three relations will be considered equivalent but since the second representation is not very efficient it will not be considered further. Another, yet similar, way of looking at a relation is as a table. Each row of the table corresponds to a tuple. The ordering of the rows is immaterial just as the ordering among the tuples in a mathematical relation is immaterial. The ordering of columns is also immaterial as long as the columns are labelled appropriately. Again we can think of the column labels as being variable names. The column label does two things. First of all it specifies what the value in that position in a tuple represents and

secondly it specifies what role the thing identified in the tuple plays in the relationship defined by the tuple.

As an example consider the following table

employee # (husband)	employee # (wife)
10001	10005
20012	21345
31456	41276

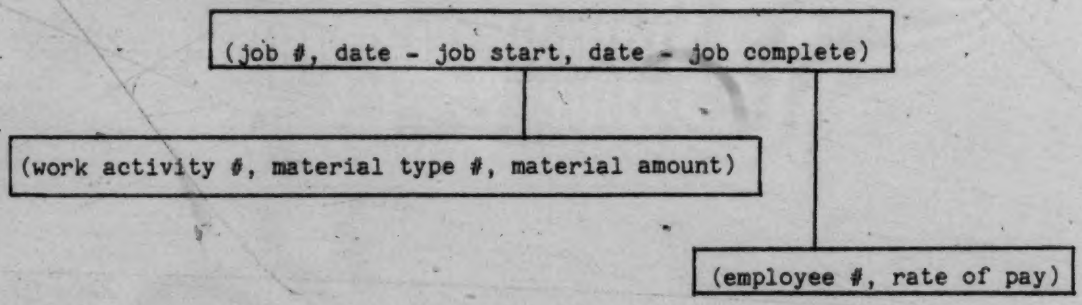
The column heading indicates first of all that the symbol 20012 identifies one of a set of people working for some company. Secondly the column heading indicates that the employee identified plays the role of husband. The table here is simply an efficient way of expressing the following statements. "The person identified by employee number 10001 is the husband of the person identified by employee number 10005". "The person identified by employee number 20012 is the husband of the person identified by employee number 21356". The statements are identical except for the employee numbers which vary from one statement to the next. As far as the set of statements is concerned the employee numbers are values of variables. This demonstrates that a table is appropriate whenever information can be broken into a set of similar statements. Another example is the following table:

employee # (person born)	day identifier (day on which birth took place)
2003	42-10-11
1015	48-09-14

In this case the symbol 1015 identifies a person having being born on the day identified in the second column. The symbol 48-09-14 identifies a day in time. The day in time on the other hand plays the role of being the day on which an employee was born.

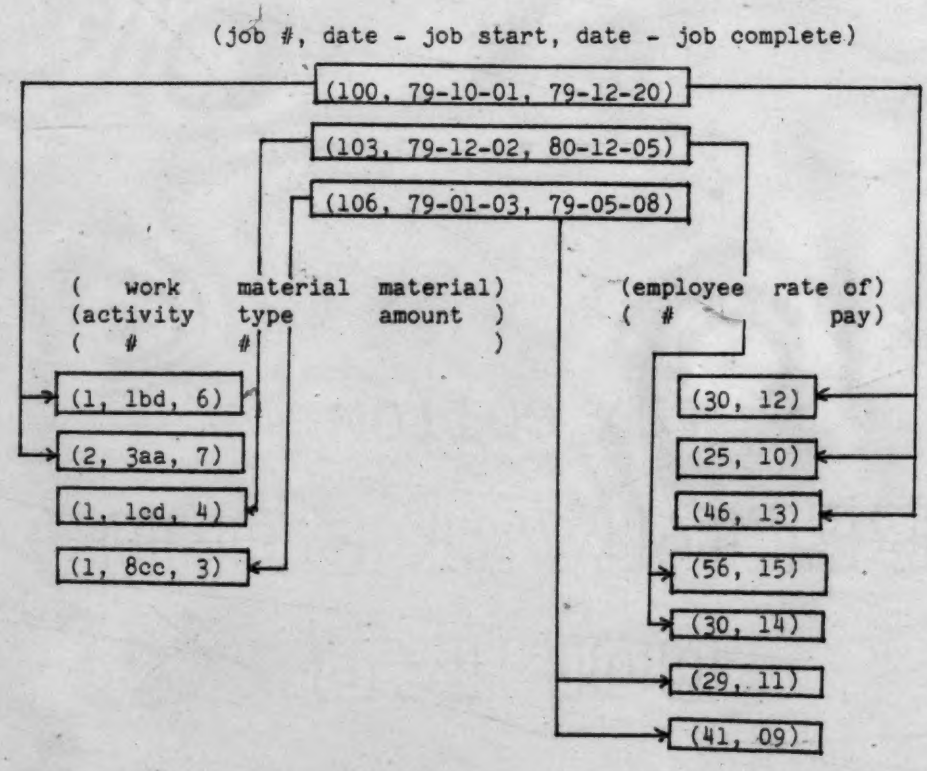
Codd's First Normal Form

After this brief comment on the meaning of relations the ideas of Codd will be presented. Codd considers the table to be the simplest form for presenting information. This must certainly be so since as indicated before, a table is simply a list of like statements. Codd's name for this type of presentation is 1st Normal Form (1NF). The following example shows how information expressed in hierarchical form can also be expressed in 1NF:



Each job has a unique "date-job start" and a unique "date-job complete". For each job there are several work activities. For a particular job each work activity is uniquely identified by "work activity #". In the set of all work activities however the work activity is only identified uniquely through the use of "job #". In each work activity only one type of material is used. Each box in the drawing above represents a set of tuples. The arrow indicates that a particular tuple of the type represented by the box at the tail of the arrow can point to 0 or more

tuples of the type represented by the box at the head of the arrow. A particular instance of the above is:



The information represented by the above can be represented by the following tables:

job #	date - job start	date - job complete
100	79-10-01	79-12-20
103	79-12-02	80-12-05
106	79-01-03	79-05-08

job #	work activity #	material type #	material amount
100	1	1bd	6
100	2	3aa	7
103	1	1cd	4
106	1	8cc	3

job #	employee #	rate of pay
100	30	12
100	25	10
100	46	13
103	56	15
103	30	14
106	29	11
106	41	9

We can also write the information in the form.

(100, 79-10-01, 79-12-20, 1,1bd,6, 2,3aa,7 30,12, 25,10, 46,13)

(103,79-12-02, 80-12-05, 1,1cd,4, 56,15, 30,14)

(106,79-01-03, 79-05-08, 1,8cc,3, 29,11, 41,9)

In this form it is difficult to see that this set of three tuples forms a relation. If we consider each value to be a component we have 15 components in the first tuple, 10 components in the second tuple and 10 components in the third tuple. Thus the set of tuples definitely do not form a relation if the individual values are considered to be components.

A similar but slightly differently organized form is:

[100,79-10-01,79-12-20,[(1,1bd,6),(2,3aa,7)],[(30,12),(25,10),(46,13)]]

[103,79-12-02,80-12-05,[(1,1cd,4)],[(56,15),(30,14)]]

[106,79-01-03,79-05-08,[(1,8cc,3)],[(29,11),(41,9)]]

Each tuple now has 5 components. The symbols [and] are used to enclose a set of elements which are tuples in this case. The first three components are single values while the fourth and fifth components are relations in themselves i.e. set of tuples. The first components are values of "job #", the second components are values of "date - job start", the third components are values of "date - job complete", the

fourth components are instances of the relation type called work activity data while the fifth components are instances of the relation type called employee data. For a relation to be in 1NF however each component in any tuple in the relation must be a single value which can be a tuple but cannot be a set of tuples.

Functional Dependency

One of the more important properties of the associations among elements of data used to represent information is that of functional dependency. One variable, called the dependent variable, is functionally dependent upon another variable, called the independent variable, in a relation if, over all the tuples in relation, with any one value of the independent variable there is associated only one value of the dependent variable. The significance of functional dependency is that if variable 2 is functionally dependent upon variable 1 in some relation, choosing a value for variable 1 will select a unique value of variable 2. In this way the value for variable 2 depends uniquely upon the value for variable 1.

Returning to the example and the relation with heading (job #, date-job start, date-job complete) the "date-job start" and the "date-job complete" are functionally dependent upon the "job #" because in that relation for any one "job #" there is only one "date - job start" and only one "date - job complete". In fact the functional dependency would hold for all instances of this relation type. It is also true for the relation in the example that "job #" is functionally dependent upon "date - job start". This however would not necessarily hold for all instances of the relation which might exist in the data base overtime, since it is

to be expected that more than one job may start on the same date. Since what is of interest is not a particular relation but rather all instances of a relation type let us specify that for one variable to be functionally dependent upon another variable the property of functional dependency defined above must now hold for all instances of a relation type. According to this more restrictive definition of functional dependency the "job #" would not be functionally dependent upon "date - job start".

The definition of functional dependency can also be extended to collections of variables. One collection of variables is functionally dependent upon another collection of variables if within all instances of a relation type a particular combination of values for the one collection of variables has associated with it only one particular combination of values for the second collection of variables. In the example preceding the previous one "material amount" is functionally dependent upon the collection of variables "job #" and "work activity #". This assumes that for each work activity only one type of material is placed. If it were possible to place more than one type of material for a work activity the "material amount" would be functionally dependent upon the collection of variables "job number", "work activity number", and "material type". On the other hand "material amount" is not functionally dependent upon any subset of this collection of variables.

Relation vs Relation Type

Before continuing discussion of the relationship model it is necessary to emphasize the distinction between relation and relation type. According to mathematics a relation is a particular set of tuples

where each tuple has the same number of components and all the first components are from one set, the second components from another set and so on. The thing of interest often, however, is the set of all relations of a certain type within some particular context. Two relations will henceforth be considered to be of one type if 1) the tuples in both relations have the same number of components and if 2) corresponding components from both relations are elements of the same sets. In other words two relations belong to the same type if the union of the relations is a relation, the union being of two sets of tuples.

From now on if a relation is being discussed the set of all relations of a particular type is intended.

Candidate Key

For a particular relation it is true by definition that each tuple is distinct. However it is also possible that a certain subset of the components of the tuples have a distinct combination of values from one tuple to the next. This combination is called a candidate key for the relation. It is a candidate key for the relation type if it holds true for all relations belonging to the type. In the relation type (job #, date - job start, date - job complete) the "job #" is a candidate key. This says that for every instance of the relation type each tuple will have a unique value for the "job #" which cannot be said for the other two data element types.

Another example of a candidate key is in the relation type (job #, work activity #, material type #, material amount). Let us assume that a work activity is identified by "job #" and "work activity #". Let us also assume however that several types of material can be placed by one

work activity. In this case the candidate key is the concatenation of "job #", "work activity #" and "material type #". A further restriction on the definition of candidate key is that no variable can be dropped from the combination making up a candidate key without losing the property that the combination of values of the candidate key uniquely identify tuples. Thus in the example although values of the combination "job #", "work activity #", "material type #" and "material amount" can be used to uniquely identify tuples the combination of variables is not a candidate key.

A property of the candidate key is that every variable in the relation is functionally dependent upon the combination of variables making up a candidate key. This is because each value of the candidate key can appear in only one tuple and each tuple has only one value of each variable. Therefore for each value of the candidate key there can be only one value of each of the variables. Another property of the candidate key is that no part of the candidate key is functionally dependent on any other part of the candidate key. To demonstrate this consider the relationship consisting of tuples of the form $(x_1,x_2,x_3,x_4,x_5,x_6)$. Let us assume that x_1,x_2,x_3,x_4 make up a candidate key. Let us also assume that x_3,x_4 are functionally dependent upon x_1,x_2 this implies that:

$x_1,x_2 \rightarrow x_3,x_4$

$x_1,x_2 \rightarrow x_1,x_2$

$x_1,x_2 \rightarrow x_1,x_2,x_3,x_4 \rightarrow x_1,x_2,x_3,x_4,x_5,x_6$

And we have the situation that x_1,x_2 uniquely determine a particular tuple. But a subset of a candidate key by definition cannot uniquely determine tuples. Therefore we end up with a contradiction and x_3,x_4

cannot be functionally dependent upon x_1, x_2.

Although there may be several candidate keys in a relation one of them is chosen to be the primary key. It must satisfy the condition that within every instance of the relation type every variable making up the primary key always has a value.

Functions vs Relations

Let us now digress for a moment to mathematics. According to mathematics a function is a relation such that no two distinct members have the same set of first components where first components includes all but the last component. As an example consider $(x_1, x_2, \text{---} x_n)$. For this relation to be a function it must be true that if there are two tuples in the set with the same values for x_1, x_2 -- x_{n-1} then also the two values for x_n must be identical which in turn means that the two tuples are identical. This of course cannot be so because a set cannot have two identical elements. It should be noted that x_n does not have to stand for a single variable but may actually be a group of variables. If we now state that ordering of values in tuples of a relation is immaterial as long as the values can be associated with the appropriate variables we can generalize the definition of a function further and state that a relation is a function as long as there is a proper subset of variables whose values uniquely identify tuples within the relation. Thus a relation is a function if there is a candidate key which does not use all the variables in the tuples and leaves some variables to be functionally dependent upon the candidate key. After this brief digression we will now continue with the work of Codd.

Codd's Second Normal Form

The Second Normal Form (SNF) defined by Codd can best be described by the following example. Consider the relation type (job #, work activity #, material type # - of material placed, quantity - of material placed, date - job start). Let us assume that there can only be placed one type of material in a given work activity and there is only one starting date for a job. A candidate key is then the combination consisting of "job #" and "work activity #". In fact it is the only candidate key and therefore also the primary key. Let us now consider a particular instance of this relation type.

(10010, 1, a, 4, 79.10.23)

(10010, 2, b, 3, 79.10.23)

(10010, 3, a, 5, 79.10.23)

(10011, 1, b, 8, 79.01.14)

The "date-job start" is identical in some of the tuples and only changes from one tuple to the next as the "job #" changes. In fact in this instance of the relation type the "date-job start" is functionally dependent upon the "job #". Based on an understanding of the information represented by the data it is expected that for all instances of this relation type the above will be true. If there are a large number of possible work activities for a particular job the "date-job start" values would be repeated a large number of times without providing additional information. It would be better to split instances of the relation type into:

(job #, work activity #, material type # - of material placed, quantity-of material placed)

and

(job #, date-job start)

Although the unsplit relation is in 1NF it is not considered to be in SNF. The two relations into which the former relation is split are however in SNF. The first relation has as the only candidate key the combination of the variables "job #" and "work activity #". Every variable is dependent upon the entire candidate key and not on a subset of the variables making up the candidate key. (called full dependence by Codd). In the second relation a candidate key is "job #" and again each non-candidate key variable is fully dependent upon the candidate key.

As another example consider the relation type (job #, work activity #, account code, material type #, quantity-of material placed, amount-job-account). Assume for this example that there are several work activities per job, several material types for each work activity and several accounts for each job. The quantity of material placed is the quantity of material for a particular material type, job and work activity combination. The only candidate key in this case is the combination of "job #", "work activity #", "account code" and "material type #". The variables which do not participate in any candidate key are therefore "quantity - of material placed" and "amount-job-account". The "quantity-of material placed" is functionally dependent upon the combination of "job #", "work activity #" and "material type #" and therefore not fully functionally dependent upon the candidate key. The "amount-job-account" is functionally dependent upon the combination of "job #" and "account code" and therefore not fully functionally dependent

upon the candidate key either. The relation is therefore not in SNF. The relation can be decomposed into the two SNF relations (job #, work activity #, material type #, material amount) and (job #, account code, amount-job-account). The first relation is a function with "job #", "work activity #" and "material type #" as independent variables and "material amount" as the dependent variable. The second relation is a function with "job #" and "account code" as independent variables and "amount-job-account" as the dependent variable.

Codd in his definition states that a relation is in SNF if every non-prime variable is fully functionally dependent upon every candidate key. A prime variable according to Codd's definition is one which participates in at least one candidate key. An example to illustrate that only non-prime variables have to be fully dependent upon every candidate key is the following:

(job #, account #, job name, amount-job-account)

Let us assume that for each job there are several accounts associated with it and the amount corresponds to a job and account combination. In this case there are two candidate keys the one being a combination of "job #" and "account #" and the other a combination of "account #" and "job name". This assumes that each job has a unique name. The only non-prime variable in this case is "amount-job-account" which is fully functionally dependent upon both candidate keys. The "job name" however is not fully functionally dependent upon the candidate key consisting of "job #" and "account #", nor is the "job #" fully functionally dependent upon the candidate key consisting of "account #" and "job name". The job name is functionally dependent upon the combination of "job #" and

"account #", otherwise the combination of "job #" and "account #" would not be a candidate key. According to Codd's definition the relation type mentioned above is in SNF. It may be observed at this point that the candidate keys within a relation must bear a one to one relationship to each other within a relation. By definition a candidate key value points to only one tuple and a tuple has in it only one value for the candidate key. Thus there is a one to one correspondence between candidate keys and tuples. Since each candidate key has a one to one correspondence to the tuple each candidate key must also have a one to one correspondence to every other candidate key in the relation. Thus in the example the candidate key consisting of "job #" and "account #" has a one to one correspondence to the candidate key consisting of "job name" and "account #" which follows from the fact that there is a one to one correspondence between "job name" and "job #" and the obvious one to one correspondence of "account #" to itself.

It may also be noticed that the relation above, although in SNF, can still be decomposed with a corresponding reduction in data without loss of information. The two relations into which it can be decomposed are (job #, account #, amount-job-account) and (job #, job name). If we have say n_1 jobs and n_2 accounts for each job then an instance of the undecomposed relation type would consist of n_1n_2 tuples with 4 values in each tuple or $4n_1n_2$ values. The number of values in total for the decomposed would be $3n_1n_2+2n_1$. A difference of $n_1(n_2-2)$ which is considerable if n_1 or n_2 are large. If we take the combination of "job name" and "account #" to be the primary key, the "job #", which is also a prime variable but not a primary key variable, is not fully dependent upon the primary key. Thus the SNF does not go far enough in

eliminating redundancy since it only requires that non-prime variables be fully dependent upon all candidate keys. A reason for not doing the decomposition is that it may be undesirable to lose candidate keys in the way that the candidate key consisting of "job name" and "account #" was lost in the decomposition.

Another property which may exist in SNF relations and imply redundancy is that of transitive dependency. Removing transitive dependence from SNF relation by further decomposition provides Third Normal Form relations which will be discussed next.

Codd's Third Normal Form

Let us consider three groups of variables called a, b, and c within some relation. If "b" is functionally dependent upon "a" and "c" is functionally dependent upon "b" then it follows that "c" is functionally dependent upon "a" also. The functional dependence of "c" on "a" is called a transitive dependency. As an example consider the relation (job #, work plan #, work activity #, material type #, price-material type). Let us assume that several work plans are possible for each job and several activities are possible for each work plan but only one material type can be placed as part of an activity. Let us also assume that there is only one price for each material type. Based on the assumption the "material type #" is functionally dependent on the combination of "job #", "work plan #" and "work activity #". Since "price - material type" is functionally dependent upon the "material type #" it is transitively dependent, and therefore also functionally dependent, upon the combination consisting of "job #", "work plan #" and "work activity #". The combination consisting of "job #", "work plan #" and "work activity

#" is therefore a candidate key. In fact it is the only candidate key. The non-prime variables are therefore "material type #" and "price-material type". In this case both non-prime variables are fully dependent upon all candidate keys and the relation is in SNF. Within any particular instance of this relation type we expect the range of material types to have fewer unique values than the number of tuples since otherwise the "material type #" would be a candidate key. (i.e. Each tuple would have a unique value for material type #.) Since the "material type #" points to a "price-material type" the "price-material type" would be repeated unnecessarily. The redundancy can be removed by decomposing the relation into the two types (job #, work plan #, work activity #, material type #) and (material type #, price-material type). Let us call the variable or group of variables through which a variable is dependent upon the candidate key an intermediate variable. The intermediate variable or group of variable cannot be a proper subset of a candidate key since then the variable in question would not be fully dependent upon every candidate key. The definition of Third Normal Form (TNF) according to Codd is: "a relation r is in the Third Normal Form if it is in SNF and every non-prime variable of r is non-transitively dependent on each candidate key of r". At this point it may be appropriate to make some more comments on the normal forms.

The purpose of the 1NF is to have a representation which can be examined for decomposition to Second and Third Normal Forms. The reason for decomposition to produce Second and Third Normal Forms is to eliminate redundancies without reducing information. If a variable within a relation is functionally dependent on something other than a candidate key the variable is dependent upon a variable or group of

variables which varies less than the candidate key i.e. the number of values in any instance of the relation type is less than the number of tuples. The relation between the dependent and independent variables can therefore be demonstrated more efficiently if separated from the rest of the relation. For example in the relation (job #, date - job start, account code, amount-job-account) the variable "date - job start" is dependent upon a variable i.e. "job #" whose range of values in any instance of the relation type is smaller in number than the number of tuples within the instance of the relation type. If a variable is not fully functionally dependent upon every candidate key then the variable is functionally dependent upon a proper subset of at least one of the candidate keys. The range of values for a subset of a candidate key must be smaller in number then the number of tuples otherwise the proper subset would be a candidate key and the other would not be a candidate key (i.e. the set of variables of which it is a subset). Again if a relation is in SNF but not in TNF there is a variable which is non-prime and functionally dependent upon a variable or group of variables which is not a candidate key and therefore varies less than the candidate keys.

Boyce - Codd Normal Form

Returning now to Codd's work let us consider again the example (job #, job name, account code, amount-job-account). This relation is in TNF although certain variables are functionally dependent upon variables with a smaller range than the set of tuples itself. The variable "job name" is functionally dependent upon the variable "job #" which has a smaller range than the combination consisting of "job #" and "account code" which is a candidate key. Similarly, the variable "job #" is functionally

dependent upon a variable i.e. "job name" which has a smaller range than the combination consisting of "job name" and "account code" which is a candidate key. It is obvious that this relation can be decomposed into the two relations (job #, account, amount-job-account) and (job #, job name) without loss of information. In the undecomposed relation there is a functional dependency such that the range of the independent variable in the function is smaller than the number of tuples in the relation. In the undecomposed relation "job name" is functionally dependent upon "job #". This relation between "job #" and "job name" can be expressed using a smaller number of tuples than the number of tuples included in the undecomposed relation. For a relation to be in TNF it is only required that all non-prime variables must be dependent on nothing less than candidate keys. Thus the undecomposed relation is in the TNF.

The Boyce-Codd Normal Form (BCNF) goes further to exclude redundancy in that no functional dependencies are included in a relation in BCNF which can be expressed or represented in a smaller set of tuples. Formally in the present writers words: a relation type is in the BCNF if every relation belonging to the type has the property that every functional dependency in the relation is such that the cardinality of the set of values of the independent variable is equal to the cardinality of the relation. This is equivalent to the definition provided in FAG-77 which states: "a relation schema r is in BCNF if, whenever a non-trivial functional dependency $x \rightarrow y$ holds in r then so does the functional dependency $x \rightarrow a$ for every column name a of r". A trivial functional dependency is the dependency which exists between a set of variables considered as the independent variable and a variable which is part of the set of variables. For example it is always true that (x_1, x_2,

x_3)-> x_2. In the definition above the fact that x-> a for every column name "a" of r implies that x is a candidate key. The definition thus states that whenever there is a non-trivial functional dependency within a relation in BCNF then the independent variables must have a range equal in cardinality to the range of the set of tuples which is equivalent to saying that the independent variables make up a candidate key. The relation type (job #, job name, account #, amount-job account) is not in BCNF since job name is dependent on job # which is not a candidate key. The relation type (job #, account #, amount-job-account) is in BCNF since the only non-trivial dependency is between "job #" and "account #" as independent variables and "amount-job-account" as the dependent variable with the independent variables constituting a candidate key.

It may be noticed that decomposing a relation to obtain BCNF relations may have the effect of removing candidate keys. Thus in the example the candidate key consisting of the combination of "job name" and "account #" is lost if the original relation is broken into the two relations (job #, account #, amount-job-account) and (job #, job name). On the other hand the definition of TNF is such that in producing a relation of TNF candidate keys need not be destroyed. This is because prime variables are allowed to be functionally dependent upon non-candidate keys.

As a more complex example consider the relation type (job #, work plan #, activity #, location identifier, date of activity, material type #, quantity of material placed). Let us assume that each tuple in the relation corresponds to a different activity which may be identified either by a combination of "job #", "workplan #" and "activity #" or by

the combination of "location identifier" and "date of activity". The latter assumes that only one activity takes place at a particular location on a given date. The functional dependencies are:

(job #, work plan #, activity #)	->location identifier
	->date of activity
	->material type #
	->quantity of material placed
(location identifier, date of activity)	-> job #
	-> workplan #
	-> activity #
	-> material type #
	-> quantity of material placed

In each functional dependency the independent variables constitute a candidate key and therefore the relation is in BCNF. Thus all functional dependencies are the result of keys.

Fourth Normal Form

Another normal form called Fourth Normal Form (FNF) described by Fagin in a paper called "Multi-valued dependencies and a new normal form for relational databases" (FAG-77) will now be discussed. The idea of multi-valued dependency plays a key role in the concept of FNF and will therefore be discussed first.

Consider the relation type (job #, date-job start, work plan #). In any occurrence of this relation type there will be a unique "date-job start" for a particular "job #" or in other words if two tuples agree in "job #" they will also agree in "date-job start". By definition the

"date-job start" is functionally dependent upon "job #". On the other hand if it is assumed that for each job there are several work plans then the same cannot be said for the relationship between "job #" and "workplan #". Intuitively it is felt however that for each "job #" there is a unique set of workplan #'s. Thus "workplan #" is dependent upon "job #" in some sense. This dependency is called multi-valued dependency. Let us now consider an instance of the above relationship type ~~i.e.~~

(1000, 79-10-05, 1)

(1000, 79-10-05, 2)

(1000, 79-10-05, 3)

(1001, 79-11-01, 1)

(1001, 79-11-01, 2)

A different relation providing the same information is

[1000, 79-10-05, (1, 2, 3)]

[1001, 79-11-01, (1, 2)]

The variables in this case are "job #", "date-job start" and "work plan #" set identifier. This demonstrates that a multi-valued dependency can be converted into a functional dependency by considering a different variable. The dependent variable is a set of values rather than single value. It should be remembered however that the latter relation is not in Codd's 1NF. This point will be discussed further when an evaluation is made of the FNF.

As another example consider the relation type (job #, work activity #, material type #, employee #, amount-rate of pay for employee on job).

An instance of this relation is

(1000, 1, a, 2000, 10)

(1000, 1, a, 2000, 12)

(1000, 1, a, 2000, 11)

(1000, 2, b, 2000, 10)

(1000, 2, b, 2000, 12)

(1000, 2, b, 2000, 11)

(1001, 1, c, 3000, 09)

(1001, 1, c, 3005, 13).

Let us now replace this relation by the relation consisting of two tuples which is not in 1NF because it does not have single values for components. i.e.

[1000,[(1,a),(2,b)],[(2000,10),(2000,12),(20001,11)]]

[1001, [(1,c)],[[(3000, 9), (3005, 13)]]

In this case it is easy to see that for each value of the "job #" variable there is a unique set of "work activity #" and "material type #" combinations. Also for each "job #" there is a unique set of "employee #" and "amount-rate of pay" combinations. Thus there is a multi-valued dependency between "job #" and the combination consisting of "work activity #" and "material type #" and a multi-valued dependency between "job #" and the combination consisting of "employee #" and "amount-rate of pay".

It may be noticed that a statement about multi-valued dependency is really a statement about independency. The relation above has the property that for a particular value of "job #" the values for the

combination consisting of "work activity #" and "material type #" are independent of the values for the remaining variables in the tuples. Also if we take a particular "job #" the range of values obtained for the combination consisting of "employee #" and "amount-rate of pay" is the same regardless of the value for the combination consisting of "work activity #" and "material type #". Thus if job # = 1000, work activity # = 1 and material type # = a, the range of the combination consisting of "employee #" and "amount-rate of pay" is [(2000, 10), (2000, 12), (2001, 11)]. If job # = 1000, work activity # = 2 and material type = b, the range of the combination consisting of "employee #" and "amount-rate of pay" is the same. The same applies if the job # = 1001. Let us now consider the relation obtained from the previous one by removing one of the tuples, say the first one. In that case the range of values for the combination consisting of "employee #" and "amount-rate of pay" for job # = 1000, work activity # = 1 and material type # = a is [(2000, 12), (2001, 11)] while for work activity # = 2 and material type # = b, the range is [(2000, 10), (2000, 12), (2001, 11)]. Therefore for this new relation it cannot be said that values obtained for the combination consisting of "employee #" and "amount-rate of pay" for a particular job number is independent of the values for the combination consisting of "work activity #" and "material type #". Therefore neither combination is multi-valued dependent upon "job #".

After these brief examples let us now consider some formal definitions. First we define independency. Within a relation two variables are independent of each other if with every value of the one variable there appears in some tuple every value of the other variable, where every value means every value in their respective ranges defined by

the set of tuples. The definition implies that in the case of two variables which are independent of each other we obtain, by considering only the components corresponding to the variables, a relation which is the Cartesian product of the two ranges corresponding to the two variables. The definition for independence between two sets of variables within a relation is similar, the difference being that the ranges are not single values but tuples. Returning to the example consider the two sets of variables: the combination consisting of "work activity #" and "material type #" and the combination consisting of "employee #" and "amount-rate of pay". The range for the first set of variables is (1, a), (2, b), (1, c) while the range for the second set of variables is (2000, 10), (2000, 12), (2001, 11), (3000, 9), (3005, 13). Since the Cartesian product of these two sets has 15 elements in it while the relation has only 8 tuples it is obvious that by definition the two sets of variables are not independent.

Let us now consider a weaker sense of independence. Two variables in a relation are independent of each other with respect to a third variable in the relation if the two variables are independent of each other in the stronger sense for each particular value of the third variable. Thus if we divide the relation into separate relations by grouping tuples with the same value for the third variable then the two variables must be independent within each of the separate relations. The definition is similar for sets of variables if we remember that values are tuples rather than single values. Returning to the example consider the three sets of variables "job #", as one variable, the combination consisting of "work activity #" and "material type #" as another set and the combination consisting of "employee #" and "amount rate of pay" as the

third set. Let us then divide the relation into separate relations according to "job #".

We obtain

(1000,1,a,2000,10)	(1001,1,c,3000,09)
(1000,1,a,2000,12)	(1001,1,c,3005,13)
(1000,1,a,2001,11)	
(1000,2,b,2000,10)	
(1000,2,b,2000,12)	
(1000,2,b,2001,11)	

If we ignore the "job #" in the first relation the relation is the Cartesian product of two sets of variables ie. [(1,a), (2,b)] x [(2000,10), (2000,12), (2001,11)]. Similarly if we ignore the "job #" in the second relation we obtain the Cartesian product [(1,c)] x [(3000,9), (3005,13)]. The relation can be written as:

(1000,[1,a),(2,b)] x [(2000,10),(2000,12),(2001,11)])

(1001,[(1,c)] x [(3000,09),(3005,13)])

In the relations above, obtained by partitioning the original relation by job #, the two sets of variables are independent of each other. With this definition of independence it is straight forward to define dependency.

A definition of multi-valued dependency can be made in terms of independency as follows. A set of variables y is multi-valued dependent upon another set of variables x if the remaining set of variables z within the relation is independent of the set of variables y with respect to the set of variables x. Since the independence is symmetrical it can be said that if y is multi-valued dependent on x and if z includes the

remaining set of variables, z is also multi-valued dependent on x.

It is now possible to define the Fourth Normal Form as proposed by Fagin (FAG-77) in whose words "a relation schema R# is in Fourth Normal Form if, whenever a non-trivial multi-valued dependency x->>y holds for R#, then so does the functional dependency x-> a for every column name "a" of R#". This means that all dependencies, multi-valued or not, must have x for the independent variables. In other words all dependencies are the results of keys. Let us again consider the example (job #, work activity #, material type #, employee #, amount - rate of pay). In this case the combination consisting of "work activity #" and "material type #" is multi-valued dependent upon "job #" as is the combination consisting of "employee #" and "amount - rate of pay". Decomposing this relation type we obtain the two relation types (job #, work activity #, material type #) and (job #, employee #, amount - rate of pay). An instance of the undecomposed relation type is

(1000,1,a,2000,10)

(1000,1,a,2000,12)

(1000,2,b,2000,11)

(1000,2,b,2000,10)

(1000,2,b,2000,12)

(1000,2,b,2000,11)

(1001,1,c,3000,09)

(1001,1,c,3005,13)

The corresponding instances of the component relations are

(1000,1,a)		(1000,2000,10)
(1000,2,b)	and	(1000,2000,12)
(1001,1,c)		(1000,2000,11)
		(1001,3000,09)
		(1001,3005,13)

The one and only candidate key for the first of the above two relations is the combination consisting of "job #", "work activity #" and "material type #" while the one and only candidate key for the second is the combination consisting of "job #", "employee #" and "amount rate of pay". Although it cannot be said for the two instances above it is true that for the two relation types of which the above are instances that there are no non-trivial multi-valued dependencies. A trivial multi-valued dependency is that within the relation (x,y) it is always true that $x \twoheadrightarrow y$. Thus the two relation types are in FNF.

Fourth Normal Form vs the Other Normal Forms

Since functional dependency is a special case of multi-valued dependency the definition of FNF implies that if a relation is in FNF it is also in BCNF. If a relation is in BCNF it is not necessarily in FNF however. Let us assume that (x_1, x_2, x_3) is a relation such that $x_1 \twoheadrightarrow x_2$ and $x_1 \twoheadrightarrow x_3$. The relation is in BCNF because it is all key but it is not in FNF and can be decomposed into the two relations (x_1, x_2) and (x_1, x_3).

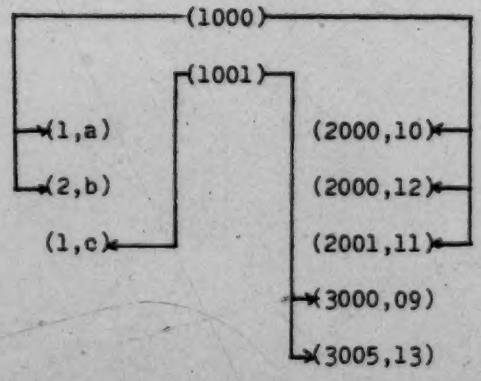
Let us now take a closer look at the FNF. The FNF implies the BCNF which implies the TNF and so on. Therefore it is usual to consider the

conversion to FNF to be the final step in decomposing a relation. A relation is in FNF if there are no strictly multi-valued dependencies and the functional dependencies are the results of keys. It is therefore common to think of a reduction in redundancy in going from 1NF, to SNF, to TNF, to BCNF and finally to FNF. Is the removal of multi-valued dependencies really a step however which follows reduction to the BCNF? Removal of multi-valued dependencies can also be seen as a step preceding reduction to 1NF. This can be illustrated by the example of the relation not in 1NF ie.

[1000,[(1,a),(2,b)],[(2000,10),(2000,12),(2001,11)]]

[1001,[(1,c)],[(3000,9),(3005,13)]]

The simplest way of converting this relation to an equivalent relation in 1NF is to simply expand each tuple into several tuples giving the relation as per example in the beginning of this section. However that is not the usual way to convert this to the 1NF. If we describe the non-normalized form in hierarchical form it immediately becomes obvious how the relation should be converted to 1NF. In hierarchical form the relation becomes:



This may be converted into the two 1NF relations

(1000,1,a)	(1000,2000,10)
(1000,2,b)	(1000,2000,12)
(1001,1,c)	(1000,2001,11)
	(1001,3000,09)
	(1001,3005,13)

These two relations are also in FNF. Thus if a relation is converted to 1NF properly the multi-valued dependencies will be removed and the relation will also be in FNF.

Multi-Valued Dependencies and Hierarchical Relations

The following is intended to show the similarity between multi-valued dependency and the hierarchical form of a relation. Consider the following diagram.

Each node in the diagram above represent a set of tuples whose components are simple values. The diagram is to be interpreted to mean that a tuple of type "a" is connected to 0 or more tuples of type "b" and 0 or more tuples of type "c". We can also think of the nodes as representing sets of variables. Thus "a" represents the set of variables a_1, a_2, --- a_n, "b" represents the set of variables b_1, b_2, --- b_m and "c" represents the set of variables c_1, c_2, --- c_n. The variables a_1, a_2, --- a_n take on values which form the components of the tuples in the relations. We can now also think of "a" as being a

variable, a value for which is the set of values for the variables a_1, a_2, --- a_n. A value of "a" is a specific tuple belonging to the set of tuples identified by the letter "a". The graph may now be interpreted in the following manner. The fact that "a" points to "b" means that a particular occurrence or value of "a" is associated with or points to 0 or more values of "b" and is associated with or points to 0 or more values of "c". This graph also indicates that a particular value of "b" is associated with exactly one value of "a" and a particular value of "c" is associated with exactly one value of "a", assuming that values of "b" and "c" are not repeated. Thus each value of "a" points to a set of values of "b" and a set of values of "c" where the sets may be empty.

The diagram clearly indicates that the set of values for "b" pointed to by a value of "a" is independent of the value for "c". Thus the variables "b" and "c" are independent of each other with respect to "a" and "b" and "c" are therefore multi-valued dependent upon "a". Thus we can also write a ->> b a ->> c.

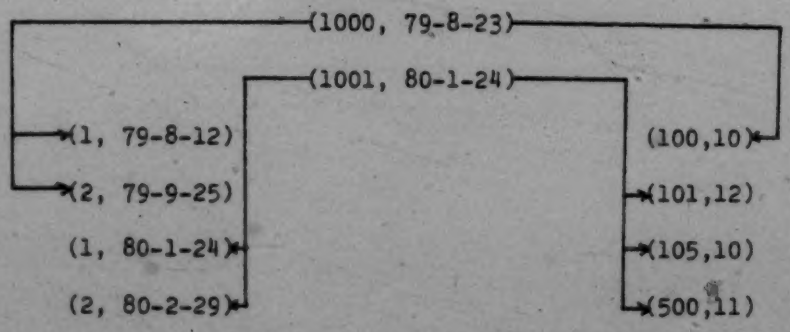
Let us now consider a particular relation having this form i.e.

a = (job #, date - job start)

b = (workplan, date - workplan start)

c = (employee #, amount - rate of pay)

with values:



This hierarchical form can be expressed by one normalized relation as

(1000, 79-8-23, 1, 79-8-12, 100, 10)

(1000, 79-8-23, 2, 79-9-25, 100, 10)

(1001, 80-1-24, 1, 80-1-24, 101, 12)

(1001, 80-1-24, 1, 80-1-24, 105, 10)

(1001, 80-1-24, 1, 80-1-24, 500, 11)

(1001, 80-1-24, 2, 80-2-29, 101, 12)

(1001, 80-1-24, 2, 80-2-20, 105, 10)

(1001, 80-1-24, 2, 80-2-29, 500, 11)

The tuples are obtained by taking a particular value of "a", a value of "b" that goes with it and all the values of "c" that go with this value of "a". Thus for each value of "a" we generate tuples by combining the Cartesian product of values of "b" and "c" that go with it.

The hierarchical relation however can also be replaced by the following two relations.

(1000, 79-8-23, 1, 79-8-12)

(1000, 79-8-23, 2, 79-9-25)

(1001, 80-1-24, 1, 80-1-24)

(1001, 80-1-24, 2, 80-2-29)

and

(1000, 79-8-23, 100, 10)

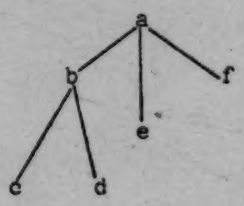
(1001, 80-1-24, 101, 12)

(1001, 80-1-24, 105, 10)

(1001, 80-1-24, 500, 11)

These two relations are a decomposition of the former relation obtained by splitting the original relation according to independency.

A slightly more complicated hierachial relation can be represented by the following graph



This graph is equivalent to

a -> b

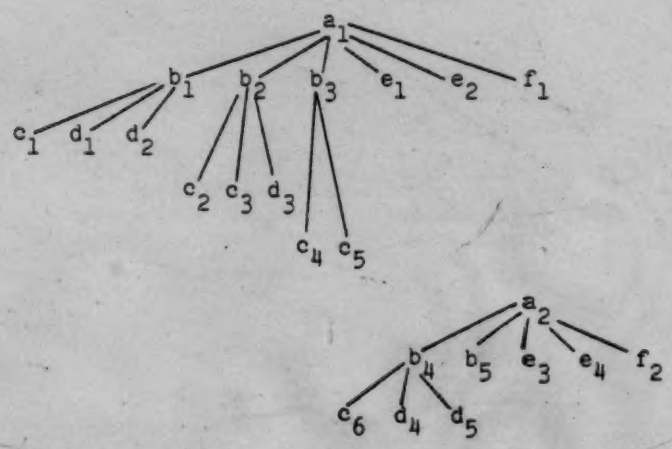
a -> e

a -> f

b -> c

b -> d

Thus "b", "e" and "f" are pairwise independent of each other for every value of "a". "C" and "d" are independent of each other for every combined value of "a" and "b". If values of "b" are not repeated then "c" and "d" are also independent of each other for every value of "b". A particular relation of the above form is



Where a_1, a_2 are values of "a"; b_1, b_2, b_3, b_4, b_5 are values of "b" and so on. It must be emphasized that a_1, a_2, etc., are constant values here rather than being variables as in a previous example. These values may in general be tuples. Replacing the above by one normalized relation we obtain:

$(a_1, b_1, c_1, d_1, e_1, f_1)$

$(a_1, b_1, c_1, d_1, e_2, f_1)$

$(a_1, b_1, c_1, d_2, e_1, f_1)$

$(a_1, b_1, c_1, d_2, e_2, f_1)$

$(a_1, b_2, c_2, d_3, e_1, f_1)$

$(a_1, b_2, c_2, d_3, e_2, f_1)$

$(a_1, b_2, c_3, d_3, e_1, f_1)$

$(a_1, b_2, c_3, d_3, e_2, f_1)$

Etc.

The hierarchical relation can also be represented by the four normalized relations as

(a_1, b_1, c_1) (a_2, b_4, c_6)

(a_1, b_2, c_2)

(a_1, b_2, c_3)

(a_1, b_3, c_4)

(a_1, b_3, c_5)

and

(a_1, b_1, d_1) (a_2, b_4, d_4)

(a_1, b_1, d_2) (a_2, b_4, d_5)

(a_1, b_2, d_3)

and

(a_1, e_1) (a_2, e_3)

(a_1, e_2) (a_2, e_4)

and

(a_1, f_1) (a_2, f_2)

This ends the explanation of the normalization process and the message model of Langefors will be considered next.

CHAPTER 6

THE MESSAGE MODEL OF LANGEFORS

Central to Langefor's model is what he calls an elementary message or e-message for short. Elementary messages are like statements or sentences. The words in the sentence are analogous to data terms. Although words do have meaning and information associated with a given sentence is dependent upon the meaning associated with the words in the sentence, the words do not carry information on their own. Thus to say "cat" does not provide any information but to say "The cat is a domestic animal" does provide information. In the words of Bar-Hillel "Nothing short of a statement conveys information, and this information is not built up piecemeal from an accumulation of the information carried by each signal ---" (BAR-65). Since information is normally communicated by means of sentences a basic understanding of information can be obtained by considering sentence structure.

A sentence is a group of words stating, asking, commanding, requesting, or exclaiming something and usually consists of a subject and predicate. The subject on the other hand is the word or group of words in a sentence about which something is said and which serves as the starting point of the action except in passive constructions. The predicate is the word or group of words that make a statement about the subject of a clause or sentence. In logic a predicate is something that is affirmed or denied about the subject of a proposition. Some examples of sentences with predicate underlined are "The wind blows". "The wind blows from the east", "John threw the ball", and "Grass is a plant". The

predicate in turn consists of an object which is a noun or substantative that directly or indirectly receives the action of a verb. For example "Give me the book". "Me" is the indirect object while "book" is the direct object.

Thus a sentence says something about an occurrence of an entity type. The entity about which something is said is the subject of the sentence. The part of the sentence which says something about the entity is called the predicate. The predicate on the other hand can either give a property of the entity which is the subject or it can indicate how the entity is related to another entity which then plays the role of object in the sentence. For example, consider the sentence "Jack owns the car with licence 146B" in which case the predicate indicates how a certain entity is related to the subject entity. In this case "car with licence 146B" is the object of the sentence. Let us also consider the complementary sentence "The car with licence 146B is owned by Jack". In this case "Car with licence 146B" is the subject while "Jack" is the object. Both sentences are equivalent semantically and provide the same information. Rather than saying that the sentence says something about some entity it would perhaps be better to say that the sentence indicates how two entities are related. Let us now consider another example i.e. "Jack weighs 100 lbs". In this case "Jack" is the subject while "weighs 100 lbs" is the predicate. It is difficult in this case to come up with a complement of the sentence as we did in the previous example unless we say "The set of objects which weigh 100 lbs has Jack as one of it's members".

So far the sentences have consisted of either a pair of entities with a certain relationship or an entity with an attribute. Something which

is implicit is the idea of time. Thus the sentence "The car with licence 146B is owned by Jack" should really say "The car with licence 146B is presently owned by Jack".

According to Langefors the basic unit of information consists of at least three parts which are concepts. They are an object concept, an attribute concept and the time attribute, or an object concept, a relationship concept, another object concept and the time attribute. In the example above one of the objects is represented by the words "The car with licence 146B". The other object is represented by the word "Jack" while the relationship is represented by the words "owned by".

In the case above the two objects are uniquely identified but this is not always so. For example consider the statement "Jack owns a red car at the present time". In this case one of the objects is uniquely identified, within some context that is, while the other object is not uniquely identified. Instead some of the non-unique properties of the second object are indicated. That is the properties of the second object are those shared by the class of objects called cars and the property of being red.

Returning now to the previous example we see that it is not so much different if we admit that the words "the car with licence 146B" also state the properties of an object i.e. the object is of type "car" and it has licence #146B. The difference is that the object is identified uniquely by its properties.

We can analyze the basic components of a message further by stating that it consists of (1) properties of an object (2) properties of a second object (3) a relationship between the two objects and (4) the time property of the relationship or (1) properties of an object (2) other

properties of the same object (3) the time property. The properties of the object may or may not uniquely be used to represent the object.

Let us now use another example to illustrate the duality between object-role and attribute. Consider the statement:

> "The person with employee number 3000 was born on the day with date Dec 1, 1944."

We can describe this message by saying that it depicts the relationship between two objects, the first being represented by the properties "person" and "employee number 3000" and the second being represented by the properties "day" and "Dec 1, 1944". We can also say however that the message depicts the attribute of an object represented by the properties "person" and "employee number 3000". The point of view we select depends strictly upon convenience and upon other messages that we want to relate to this message. If we have other messages which say something about particular days then obviously the day with date Dec 1, 1944 should be thought of as representing an object.

To see how it is sometimes simpler to consider object-relationship pairs rather then attributes let us assume we are dealing with the object types persons, days, weeks, and years with roles of birth, death and marriage. What we have is 4 object types and 3 role types or 7 concepts. Each of the object types days, weeks and years can be combined with each of the roles birth, death and marriage giving 9 attributes or a total of 10 basic concepts. This demonstrates the factorizing effect we get by considering objects and roles instead of attributes whenever possible.

Let us now consider some more terms defined by Langefors. So far we have explained what is meant by an e-message or elementary message. An e-message on the other hand is an instance of what Langefors calls an e-concept. An e-concept is another word for e-message type. Up to this point we have only considered information and although we cannot discuss information without some form of representation it is not the representation that has been the focus of our interest.

For an e-message to be communicated a representation is required. This representation is called an e-entry by Langefors. The term e-entry is used as opposed to e-record because the word record suggests a contiguously stored set of symbols. A set of e-entries is called an e-file.

The importance of the concept, defined by Langefors, lies in the fact that it recognizes the various stages of systems development. During the initial stages the e-messages are considered without regard for representations. During the following stages representations for the e-messages are decided upon. These representations are still of importance to the user. During the final stages of development representations for computer storage are to be considered. Langefors considers at least two models of the data base. One is called the infological model while the other is called the datalogical model. In the infological model only information is considered while in the datalogical model the representation of the information is considered. That is, the representation is considered without regard for storage.

Having discussed some of the more common models for information and/or data analysis we will next consider a variation of these as proposed by this author.

CHAPTER 7

A PROPOSAL FOR ANOTHER MODEL: THE ENTITY ROLE MODEL

A model or method for describing information proposed by this author will now be considered. For want of a better term this model may be called the Entity Role Model.

The main characteristic of the Entity Role Model is that greater stress is placed upon definition of entity types and of the roles the entity types play in relationships. This means that whereas someone else might interpret the definition of an entity playing a certain role in a relationship to be a property of the other entity, this model recognizes the symmetry of a relationship. As an example consider the information type which states that a certain person is born on a certain day. This information type might have the following occurrence.

person = jack - day = 50.03.01
(person born) (day on which person was born)

The information type of which the above is an example is described or viewed by the Entity Role Model as follows. The information consists of defining an association between an occurrence of two entity types. The occurrence of the first entity type plays the role of being born while the occurrence of the second entity type plays the role of being the birth date. On the other hand in a model stressing properties of entities the information would be described as follows.

person = jack - birthdate = 50.03.01

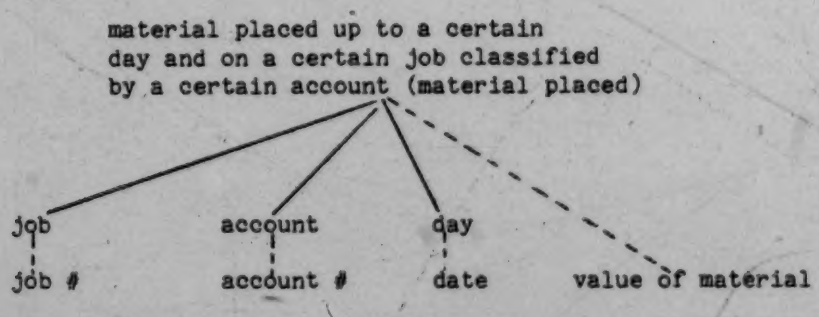
In this case the person is considered as an entity while the birthdate is considered as an attribute or property.

Let us now analyze the example further and look at various aspects of the occurrence of the message type or information kind. We will then see that the idea of attribute still plays a part in the Entity Role Model. In the example the following parts are present:

person	- label for an entity type
jack	- an attribute of the occurrence of the entity type
day	- label for an entity type
50.03.01	- an attribute of the occurrence of the entity type
person born	- a role descriptor
day on which person born	- a role descriptor
person name	- name of an attribute type
date	- name of an attribute type

This author had great resistance to considering person name as being the label of an attribute type and similarly with date. A survey of the literature also indicates some disagreement on this point. Some authors consider the value "jack" to be a surrogate for an occurrence of the entity type person. A compromise is made here by considering the person name to be an artificial attribute, a special kind of attribute or identifier attribute. Its purpose is to link information about the entity type to other information. To state only that a person's name is jack carries no useful information.

To further illustrate the points just made let us now consider a somewhat more complex example. The solid lines in the diagrams connect entity types while the broken lines connect attribute types to entity types.



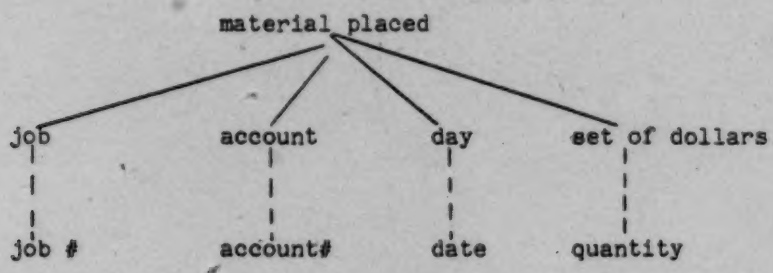
The entity types are material, job, account and day. The attributes are job #, account #, date and value of material. The roles that the entity types' occurrences play (N.B. not the attributes) are:

material placed --- job - the job, part of which the material was placed

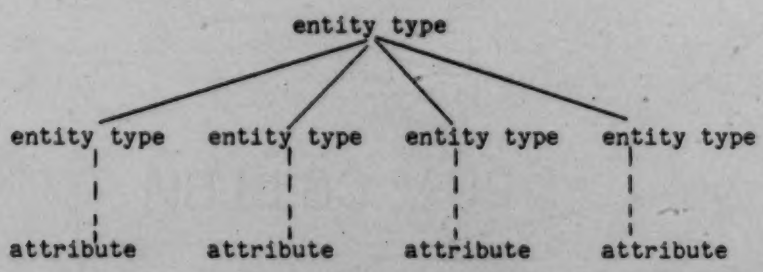
material placed --- account - the account which classifies the placement as an investment

material placed --- day - the day up to which the material was placed

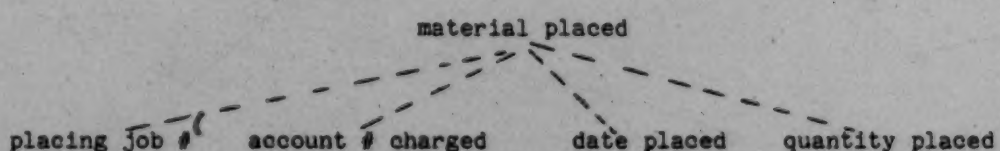
It is interesting to note that a value of material can also be considered as a property of a set of dollars which is related to the material placed entity. In this case the set of dollars is considered to be an entity type. The value of material may then be considered as an attribute of the set of dollars and may be called quantity. i.e.



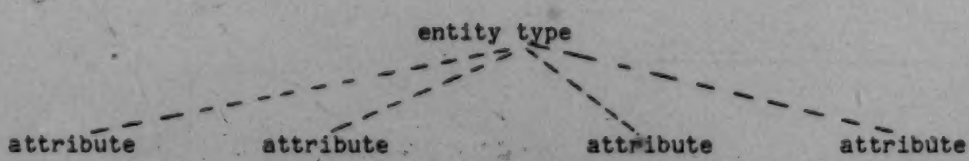
Another model of the message above is:



The four attributes now are "job #", "account #", "date" and "quantity". It may be noted here that it is the attributes which are usually considered to be the data element types. Another slightly different point of view which is still correct is:



A model of this message type is:



Thus the attributes "job #", "account #", "date" and "quantity" although basically attributes of job, account, day and set of dollars

respectively have become direct attributes of the material placed entity type.

Now then what is the significance of the two points of view. Taking one of the branches in the tree of the first point of view we have

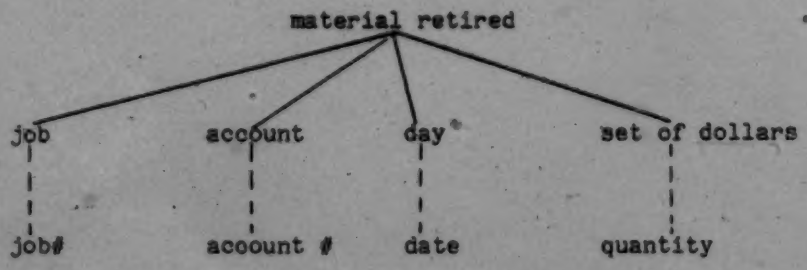
material placed ——job—— job#
(placing)

The equivalent branch in the tree for the second poing of view is

material placed------placing job #

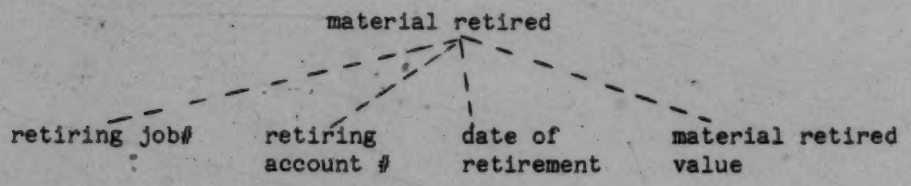
Thus job---------job# is equivalent to placing job #.
(placing)

The attribute type "placing job #" is factorized into a) an entity type b) a role for the entity type and c) an attribute for an entity type on its own. The desirability of the first point of view is that entity types are more fundamental and are definite parts of the object system which the information system is to be a model of. The attribute values in this case are made up and are really not part of the object system. Their sole purpose is to permit us to talk about the object system. Let us now consider a slightly different example, i.e.



The attribute types in both this example, and the previous example, considering the first point of view, are the same. If we take the second

point of view we get for this example



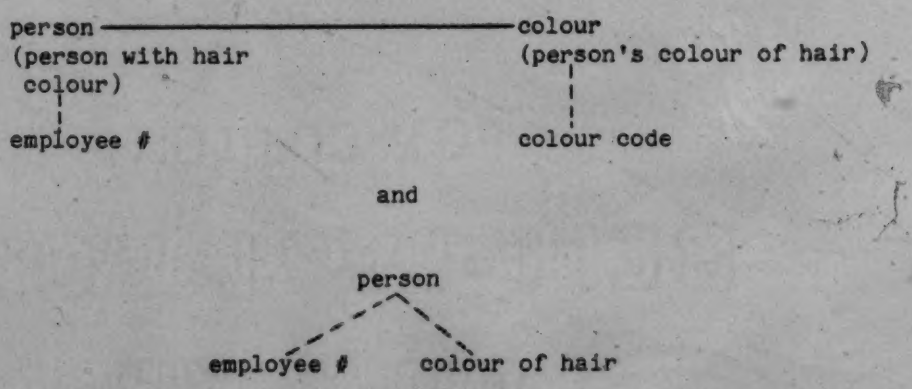
Thus if we accept the second point of view we end up with four additional attributes or four additional data element types and we don't explicitly recognize all the various entity types involved. The first point of view is somewhat extreme however in that the entity type "set of dollars" is explicitly mentioned. In this case it is perhaps best to drop mention of "set of dollars" and replace it and its role description by the attribute type name "value of material".

The advantages of the first point of view include all the advantages associated with distinguishing between objects in the object system, which exist whether or not we are interested in modelling them, and things used to represent the objects in the object system. If we think of the object system as being a set of objects with relationships among the objects then the point of view which considers entity types and role types is much more basic than the point of view considering entity types and attribute types.

It now becomes more clear regarding what has to be done in developing the conceptual model.

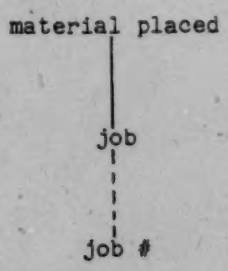
The first step is to determine the entities which are included in the Object System, not all entities of course but only those entities of interest. The next step is to classify these entities into entity types and to define each entity type. It is not expected that this step can be completed without going on to the next step which involves defining the

roles that occurrences of entity types play in association with occurrences of other entity types. Following this step the attribute types of entitities must be identified. Carrying out this step may lead back to the first step since judgement has to be used to determine whether an attribute type should be described or whether an entity type role should be described. How to make this decision can best be illustrated by the following examples:



Whether we adopt the first or second point of view depends on whether we want to link various data about colour. That is, will colour be of interest in its own right. If not, colour should not be considered as an entity type. We may not be interested in things about colour in the same way that we are interested in things about the person. Notice again the difference between the two attribute types in the second point of view. The "employee #" may be considered as an artificial attribute which is not really part of the object system. The "employee #" may also be thought of as a linking attribute. It is an attribute assigned to the entity type "employee" to permit linking of data about the employee.

Considering the example at the beginning of this chapter:



The "job #" is an attribute of the job which plays the role of having placed the material. This attribute is rather meaningless except that the job has other attributes and plays different roles in other associations and permits data about the job and other entities related to this job to be linked together. The question which now arises is whether or not the "job #" should be considered as part of the object system. This question is equivalent to "Can the entity exist without the job #?" i.e. "Does the job # exist whenever the job exists?" In this case the answer is yes and therefore the attribute "job #" may be considered as part of the object system and may be defined at the time the object system is described. The "job #" may be called an identifying attribute. Let us next consider a formal notation for describing message types.

The first step in what we may call information component analysis as opposed to information precedent analysis is to classify the entities of interest into types and to identify (label) and define each type. As indicated before it is difficult often to distinguish between an entity-role situation and an attribute situation. This is because there is no absolute distinction and depends upon one's point of view. An entity is something about which we want to say something. It appears as a subject in a sentence which represents some information unit. An attribute is something about which we don't want to say something.

Each entity type is given a descriptive name, a definition and a label which may be used in a shorthand notation. In this discussion we will consider labels of the form E_1, E_2, E_3, ----. For example:

> E_2 - Job - A set of activities planned and carried out as a unit for the purpose of constructing telephone equipment

The next step consists of defining the information units which implies describing what types of things are said about which entities. As mentioned before we say something about an entity either by indicating how another entity plays a certain role in relation to the entity or what attribute the entity has. Thus in addition to entity type definitions we also required role type descriptors and attribute definitions. A role description consists of an English phrase describing the role and a label of the form R_1, R_2, ----. For example:

> R_1-which includes
> R_2-of approval of

A description of an attribute consists of a label, a descriptive name and a narrative description. The label is of the form A_1, A_2, ---- where A stands for attribute. The labels of the entity types, the role types and the attributes may then be used to provide a formal description of a message in the form:

$$A_i E_j R_k E_q A_p$$

which reads "attribute A_i of entity type E_j in role R_k in relation to entity type E_q with attribute A_p".

As an example consider the coded message description "$A_1E_1R_1E_2A_1$" with definitions:

A_1 - identifier of

E_1 - job

E_2 - employee

R_1 - supervised by

So that an expanded description of the message type becomes: "identifier of job supervised by employee with identifier of". Notice that nothing is said about how the job is identified in an actual occurrence of the message type. The significant thing during information analysis is that the message type is uniquely identified. It is during data analysis that different coding schemes should be considered. A slightly different example is the message type "$A_1E_1R_1E_2A_2$" with the same components as before except that A_2 means "age of". A description of the message type then becomes "identifier of job supervised by employee age of." Thus a specific occurrence of this message type would indicate the age of the employee supervising a particular job or conversely would indicate a particular job supervised by an employee of a certain age. So far the examples have been relatively simple.

A slightly more complex example is:

$A_1E_{23}R_1E_{21} \cdot A_1E_{24}R_{29}E_{21} \cdot A_1E_5R_1E_{21} \cdot A_{10}E_{21}$

E_{23} - Workplan Location - A point on a work plan associated with a job

E_{21} - Material Placement Work Activity - A work activity which is part of a job and takes place at one work location

E_{24} - Material Unit Type	-	A type of material unit
E_5 - Account	-	A classification of investment
A_1 - Identifier	-	An attribute of entities which together with the entity class name may be used to identify an entity
A_{10} - quantity of material	-	the amount of material placed
R_1 - which includes		
R_{29} - of material placed by		

The . can be interpreted as a concatenation operator meaning "and". The coded description of the message type can then be mechanically translated to read:

> identifier of work plan location which includes material placement activity and identifier of material unit type of material placed by material placement activity and identifier of account which includes material placement activity and quantity of material of material placement activity

The main reason for coding the message type description in this form is that it breaks the message type description into entity types, role types and attribute types. This is useful for two reasons (1) it forces the analyst to think precisely about the component parts of a message type description and (2) this coded message description can easily be analyzed by mechanized procedures.

If the attribute definition, entity type definitions, role descriptions and coded message type descriptions are stored in a mechanized file a computer program can be written to (1) translate coded message description into English narrative (2) determine which message types say something about a particular entity type and thereby indicate what type of information is provided about a particular entity type and

(3) do completeness checks to determine if every entity type appears in some message. A complete example of message type descriptions as related to a job package which contains all information regarding a job is contained in the appendix.

The next step which takes us from information analysis to data analysis is to consider how occurrences of message types are to be represented by data. It is possible for occurrences of message types to be represented in many ways. This is partly because occurrences of entity types may be identified in several ways. Thus a job may be identified by a number or by a non-numeric name. In fact it is feasible for jobs to be identified by several different numbering schemes. For each entity type we have to define the various ways that occurrences may be identified and for a particular message type decide which identifier is to be used. This activity may be called data element definition. Present practices of information system development usually use this step of data element definition to both describe the information and also the representation of the information without making a distinction between these two aspects and consequently doing a poor job of both.

Names for entity identifier, in order to be meaningful, should consist of two parts which are (1) a name for the entity type and (2) a name for the identification-scheme used. Examples are "job number", "job name", "account number", "account name", "day julian" and so on. For the purpose of documentation these may be assigned labels such as D_1, D_2, ----. Note that in the examples the name of the attribute involved is left out and the attribute is assumed to be the identifier attribute. Another example of an appropriate data element type name is "material placement work activity - material amount". The entity type name is

"material amount". The name for the representative scheme is left out because there is only one. With each coded message description we can associate a coded representation description. This is illustrated in the example below:

$$A_1E_{23}R_1E_{21} \cdot A_1E_{24}R_{29}E_{21} \cdot A_1E_5R_1E_{21} \cdot A_{10}E_{21}$$

$$D_1 \cdot D_2 \cdot D_3 \cdot D_4$$

D_1 - Work Plan Number	- a number used to identify a workplan location
D_2 - material unit type number	- a number used to identify a material unit type
D_3 - account code	- a number used to identify an account
D_4 - material placement work activity material amount	- the amount of material quantity placed

It may be noticed that the expression $D_1 \cdot D_2 \cdot D_3 \cdot D_4$ with the definitions of D_1, D_2, D_3 and D_4 on their own do not describe the information represented and therefore must be associated with the coded message description. This demonstrates where present methods of information analysis go wrong when attempts are made to define information content strictly through data element definitions. Often the data element definition turns out to be a mixture of attempts at true information type description and true representation description.

We will next consider a comparison of the various methods of information and/or data analysis.

CHAPTER 8

A COMPARATIVE DISCUSSION OF THE VARIOUS MODELS WITH REGARDS TO THEIR ROLE IN INFORMATION ANALYSIS.

The models to be discussed in this section in terms of advantages and disadvantages are Chen's Entity Relationship Model, Codd's Relationship Model, the Message Model of Langefors and the author's Entity Role Model.

Chen considers four levels of views of data (1) Information concerning entities and relationships which exist in our minds (2) Information structure - organization of information in which entities and relationships are represented by data (3) Access - path - independent data structure - the data structures which are not involved with search scheme, indexing schemes, etc. (4) Access - path - dependent data structure. Although this author does not disagree that there are at least four views of data it is difficult to understand the differences intended by Chen. If by level 1 is meant the conceptual model or a description of the concepts, by level 2 the manner in which the concepts are represented by symbols and level 4 the symbols and relationships between symbols stored in the computer then what is meant by level 3? Perhaps level 2 should be called a data structure rather than information structure since the data aspect rather than the information aspect is stressed.

Chen also considers attributes and value sets but is not able to explain the meaning of value sets precisely. It is not clear when two value sets are equal. Chen states that 12 in the value set inches is the

same as 1 in the value set feet. He considers the customer ship-to-addresses to be the same value set as the order ship-to-addresses. This case is simple because values in both sets are labels for the same entities. He also considers quantity ordered and quantity outstanding of a line item to have the same value sets called quantity. This would imply that all quantities make up one value set. An example of where Chen considers two value sets to be different is credit limit of customer and balance in account of customer. Both are presumably quantities of dollars yet Chen considers two different value sets. It is the author's opinion that value sets because of their ambiguous nature are less than useful.

Chen does not make clear his distinction between entities and attributes either. He considers date to be an attribute of an event rather than an attribute of a day which is an entity type.

Perhaps a greater weakness of Chen's model is that it is graphical and therefore does not lend itself easily to mechanized analysis. It is good only if the information to be modelled is relatively simple.

The major advantage of the Relational Model is that it is formal and relates information representation to mathematical functions. It is therefore excellent for the data model stage and good for modelling the data base. Since a relation is a set of tuples and a tuple is like a message it is not useful for analysing single messages but only for sets of like messages. The many to one or many to many relationship between data elements only makes sense when we talk about more than one tuple. One of the ambiguous areas in the Relational Model is how to join relations on a common component. Codd distinguishes the elements in a particular relation by naming a domain and role for each element in a

tuple but does not describe how elements may be named so that it becomes automatic to determine if two relations can be joined based on a common element. Elements overall are distinguished by using a relation name, domain name and role name. The Relational Model does not really deal with the problem of mapping from the information description to the data representation. It is only when the data representations are in the form of tuples which can be categorized into sets of tuples which are relations that Codd's theory applies. Codd's relation theory therefore applies more to the data model than the information model.

The Message Model of Langefors is useful in that it describes the basic requirement for information. It points out that the basic unit of information cannot be that which is represented by one data element. It tells us how to break information up into elementary messages. These elementary messages are however too small and to describe all the information produced by a system in terms of elementary messages might be too complex. Whereas Codd's theory is a theory about collections of like messages Langefors talks about single messages.

Thus the major contribution of the Message Model of Langefors is that it gives us a better understanding of the basic unit of information. It tells us how far we can go when breaking information down into its basic component. It does not tell us however what we can do in representing the information produced by a system. It provides no tool for describing information sets which are made up of these basic units of information. Thus its major disadvantage is that it stops at how information sets are analyzed or broken down and does not provide guidance for documenting information sets in total.

A major advantage of the Entity Role Model is that it factors attribute types into an entity type, a role for the entity type and an attribute for the entity type. Factoring makes it easier to define our information units. Thus instead of defining "retirement job" and "placement job" separately we define "job" first and then define "retirement" and "placement" as role type identifiers. If say we had 4 role types and 4 entity types and each role type could occur with each entity type, without factoring, we would have 10 things to define rather than 8. Entities are also more basic to our understanding. A system is often defined as a set of elements with relationships between the elements. The information systems we are talking about here are used to monitor other systems. Thus the information in the information system consists mainly of information about elements and their relationships. These elements are then the entities that we are talking about.

Another advantage of the Entity Role Model is that it lends itself to mechanized analysis. It is possible to determine what information is provided about which entities and so on.

CHAPTER 9

SUGGESTIONS FOR FURTHER RESEARCH

So far the only step that has been carried out is development of a method for describing the information to be processed by an information system. Design of a data processing system requires a data base to be designed however. What is still required is a formal method for the design of the data base.

To accomplish the design of the data base, say a hierarchical data base as is used by IBM in their IMS software, the Entity Role Model may be used as the first step. The normal form procedures can then become the intermediate steps and one of the final steps would be to combine FNF relations into hierarchical data bases, relational data bases or network data bases.

To go from the Entity Role Model to the Normal Forms requires one to think of sets of like messages and to replace the message type descriptors by combinations of variable names into tuples of the form $(x_1, x_2, -- x_n)$. These variable names are commonly called data element names. Assignment of variable names to the components of a message type descriptor has to be done with care since it effects going from the FNF to the design of the data base. Normal Form processes are essentially decomposition processes but design of the data base requires joining FNF tuples. Joining is done on common data elements or variables and it must be clear when two variables are the same. Thus further work is required to determine how variable names should be assigned to components of message descriptors.

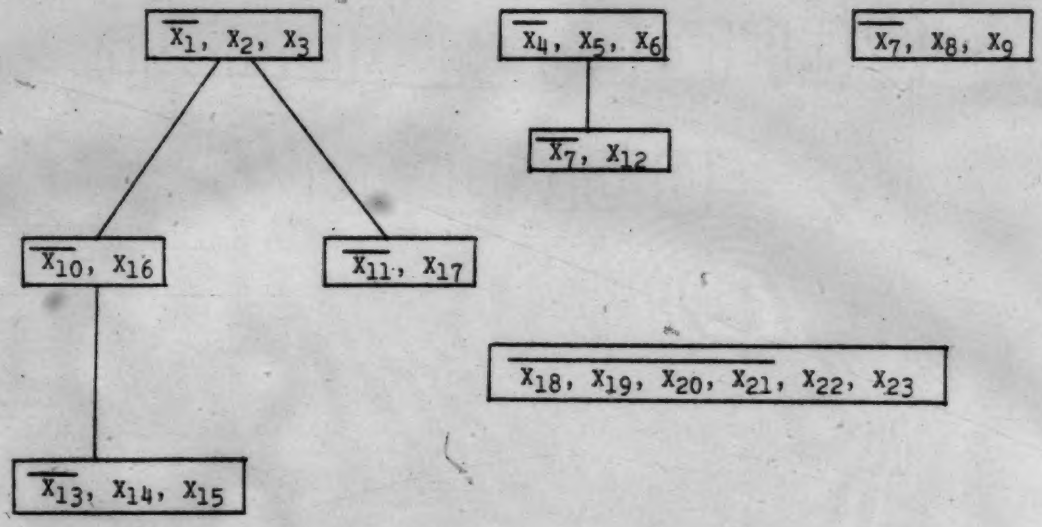
Once the FNF's have been generated the next step may be made somewhat more mechanical, at least in its preliminary stages. The success of this step will depend to a great deal on how well the variables have been named since it involves the linking of tuples into a structure. The following contains some suggestions for designing a hierarchical data base from FNF relations.

Let us say that a FNF relation is of nth degree if its primary key consists of n variables. Create a root segment for each relation of degree one and place in the root segment the primary key and all the variables which are functionally dependent upon the primary key. Also create a root segment for a relation of degree two if neither variable is a primary key on its own. If a relation is of degree two and one of its variables is already a primary key for a relation of degree one create a child segment whose parent segment is a root segment. If a relation is of degree 3 determine if two of the variables form a primary key for another relation. If so create a dependent segment. As an example consider the following set of tuples:

$$(\overline{X_1}, X_2, X_3);\ (\overline{X_4}, X_5, X_6);\ (\overline{X_7}, X_8, X_9)$$
$$(\overline{X_1, X_{10}}, X_{16});\ (\overline{X_1, X_{11}}, X_{17});\ (\overline{X_4, X_7}, X_{12})$$
$$(\overline{X_1, X_{10}, X_{13}}, X_{14}, X_{15})$$
$$(\overline{X_{18}, X_{19}, X_{20}, X_{21}}, X_{22}, X_{23})$$

The parts with the bar on top are the primary keys. A hierarchical data base consisting of these relations would look like the one following. A root segment is created for each of the three relations with X_1, X_4 and X_7 as primary keys. Next the relations of degree 2 are examined to determine which ones have X_1, X_4 or X_7 as part of their primary

keys. Each of the appropriate relations is then linked to the appropriate root segment. Note that (X_4, X_7, X_{12}) can be linked either to (X_4, X_5, X_6) or to (X_7, X_8, X_9). This takes care of all the relations of degree 1 and 2. Next the relations of degree 3 are examined. The only relation is (X_1, X_{10}, X_{13}, X_{14}, X_{15}) whose primary key consists of two sub keys which are already represented. This relation becomes the dependent of the relations with primary keys X_1 and X_{10} respectively. Next the relations of degree 4 are examined. The only one is (X_{18}, X_{19}, X_{20}, X_{21}, X_{22}, X_{23}). Its primary key cannot be decomposed into primary keys and therefore becomes a root segment on its own. In hierarchical form we get:



The process of designing the hierarchical data base is essentially the reverse of converting a set of hierarchical data bases to Normal Form. As may have been noticed however the process is not entirely mechanical at this point since certain decisions have to be made. In the above for example it is not clear whether the root segment with X_4 or

the root segment with X_7 should have the child segment with X_{12} as the functional dependent variable. Thus design of the hierarchical data base from FNF tuples requires further research as does the method for assigning variable names to the Entity Role Model components. It may be noted that the design method does not take into account access frequences.

A more complex example demonstrating the design of the hierarchical data from FNF is provided in Appendix 2.

BIBLIOGRAPHY

ABR-74: Abrial, J.R. Data Semantics. In: Data Base Management 1974. Edited by J.W. Klimbie and K.L. Koffeman. North Holland Publishing Co.

ADI-76: Adiba, M., Delobel, C., Leonard, M. A Unified Approach for Modelling Data in Logical Data Base Design. In: Modelling in Data Base Management Systems 1976. Edited by G.M. Nijssen. North Holland Publishing Co.

ANT-65: Anthony, R.N. Planning and Control Systems: A Framework for Analysis. Boston: Division of Research, Graduate School of Business Administration, Harvard University 1965.

ARM-74: Armstrong, W.W. Dependency Structures of Data Base Relationships In: Information Processing 74. North Holland Publishing Co. pp 580-585.

ASI-76: Astrahan M.M. System R: Relational Approach to Data Base Management. A C M Transactions on Database Systems. Vol 1 #2 June 1976 pp 97-137.

BAR-65: Bar-Hillel, Y. Language and Information, Selected Essays on their Theory and Application, Addisson-Wesley Publ. Co., London.

BEN-76: Benci, E., Bodart, F., Bogaert, H., Cuhanes, A. Concepts in the design of a Conceptual Schema In: Modelling in Data Base Management Systems 1970. Edited by G.M. Nijssen. North Holland Publishing Co.

BER-75: Bernstein, P.A., Swenson, J.R., Tsichritzis, D.C. A Unified Approach to Functional Dependencies and Relations. Proceedings ACM-SIGMOD Conference on DBM. May 1975.

BER-76A: Bernstein, P.A. Comment on "Segment Synthesis in Logical Data Base Design." IBM Journal of Research and Development Vol. 20, #4, July 1976, pp 412.

BER-76B: Bernstein, P.A. Synthesizing Third Normal Form Relations From Functional Dependencies ACM Transactions on Data Base Systems Vol. 1, #4, December 1976, pp 277-298.

BIL-77: Biller, H., Neuhold, B.J., Concepts For the Conceptual Schema. In: Architecture and Models in Data Base Management Systems 1977. Edited by G.M. Nijssen. North Holland Publishing Co.

BIL-78: Biller, H., Neuhold, E.J. Semantics of Data Bases: The Semantics of Data Models. Information Systems Vol. 3 1978 pp 11-30.

BLU-69: Blumenthal, C. Management Information Systems: A Framework for Planning and Development. Englewood Cliffs, N.J. Prentice-Hall, Inc. 1969.

BON-76: Bonczek, R.H., Holsapples, W., Whinston, B. Extensions and Corrections For the Codasyl Approach to Data Base Management. Informations Systems Vol. 2 1976 pp 71-77.

BOR-76: Bordewin, P., Schumacher, G., Nimsen Girl. Een Modern Database System Plus User Language ??? Vol. 18 #10 1976 pp 550-616.

BOS-62: Bosak, R. et al An information Algebra. Communications of the ACM #4 April 1962 pp 190-204.

BOY-74: Boyce, R.F., Chamberlin, D.D., King, W.F., Hammer, M.M. Specifying Queries on Relational Expressions In: Data Base Management 1974 Edited by J.W. Klimbie North Holland Publishing Co. 1974 pp 169-177.

BRA-74: Bracchi, G., Fedeli, A., Paolini, P. A Multilevel Relational Model for Data Base Management Systems. In: Data Base Management 1974. pp 211-223.

BRA-76: Bracchi, G., Paolini, P., Pelagatti, G. Binary Logical Associations in Data Modelling. In: Modelling in Data Base Management Systems 1976 pp 125-136.

BUB-76: Bubenko, J.A. Information Analysis and Data Base Design. In: Data Base Technology 1976. On Line Conferences Limited. Uxbridge, England. pp 1-20.

BUR-76: Berg, J.L. Data base directions - the next steps. Database (USA) Vol. 8 #2 Fall 1976 pp 1-49.

BUT-76: Butler, D. A Plain Man's Guide to the Relational Data Base. In: Data Base Technology 1976. pp 21-34.

CHE-76: Chen, P.S. The Entity-Relationship Model. Toward a Unified View of Data. ACM Transactions on Data Base Systems Vol. 1, #1, March 1976.

CHE-77: Chen, P.S. The Entity-Relationship Approach to Logical Data Base Design. Q.E.D. Information Sciences Inc. 141 Linden Street.

COD-62: Codd, E.F. An Information Algebra. Communications of ACM Vol. 5, #4, April 1962, pp 190-204.

COD-70: Codd, E.F. A Relational Model of Data For Large Shared Data Banks. Communications of ACM Vol. 13, #6, June 1970, pp 377-387.

COD-71A: Codd, E.F. Further Normalization of the Data Base Relational Model. In: Data Base Systems 71. Courant Computer Science Symposium 6, 1971, May 24-25, pp 33-64.

COD-71B: Codd, E.F. Normalized Data Base Structure: A Brief Tutorial. In: Proceedings of 1971 ACM - SIGFIDET Workshop "Data Description Access and Control".

COD-71C: Codd, E.F. A Data Base Sub language Founded on the Relational Calculus In: Proceedings of 1971 ACM - SIGFIDET Workshop "Data Description Access and Control". pp 35-68.

COD-74: Codd, E.F. Recent Investigations in Relational Data Base Systems. In: Information Processing 74. North Holland Publishing Co.

DAT-71A: Date, C.J., Hopewell, P. Storage Structures and Physical Data Independence. In: Proceedings of 1971 ACM - SIGFIDET Workshop "Data Description Access and Control" pp 139-168.

DAT-71B: Date, C.J., Hopewell, D. File Definition and Logical Data Independence. In: Proceedings of 1971 ACM - SIGFIDET Workshop "Data Description Access and Control" pp 117-138.

DAV-77: Davis, B. User Experience with Database Management Systems in the U.K. Database J. (GB) Vol. 7, #3, 1977, pp 2-8.

DEA-65: Dearden, J. How to Organize Information Systems. Harvard Business Review, Vol. 43, No. 2 (March - April 1965) pp 65-63.

DEL-73: Delobel, C., Casey, R.G. Decomposition of a Data Base and the Theory of Boolean Switching Functions. IBM Journal of Research and Development, September 1973.

DEL-78: Delobel, C. Normalization and Hierarchical Dependencies in the Relational Model. ACM Transactions on Database Systems Vol. 3, #3, September 1978, pp 201-222.

DUR-70: Durchholz, R., Richter, G. Information Management Concepts (IMC) For Use with DBMS Interfaces. IN: Modelling in Data Base Management Systems. Edited by G.M. Nijssen. 1970.

DUR-74: Durchholz, R., Richter, G. Concepts for Data Base Management Systems. In: Data Base Management 1974 Edited by J.W. Klimbie and K.L. Koffeman pp 97-122.

DUR-75: Durchholz, R., Relation Representation by Tables and by Functions. Information Systems, Vol. 1, 1975, pp 91-96.

ELL-78: Ellis, H.C. Developing the Conceptual Data Model to Show the Way Data is Used. Computer Bulletin December 1978.

ERI-72: Ericksen, S. The Data Base Concept. Scandanavian Honeywell Bull Magazine.

FAG-77: Fagin, R. Multivalued Dependencies and a New Normal Form for Relational Data Bases. ACM Transactions on Data Base Systems Vol. 2, #3, September 1977, pp 262-278.

FAL-76: Falkenberg, E. Significations: The Key to Unify Data Base Management. Information Systems. Vol. 2, 1976, pp 19-28.

FAL-77: Falkenberg, E. Concepts For the Coexistence Approach to Data Base Management. In: International Computing Symposium 1977. Edited by E. Morlett and D. Ribbens. pp 39-50.

FOR-61: Forrester, J. Industrial Dynamics MIT Press 1961.

GAL-77: Galbraith, J. Organization Design Addison-Wesley Publishing Company Inc. 1977.

GOR-60: Gorry, G. and Scott-Morton, M.S. A Framework for Management Information Systems - Sloan Management Review, Vol. 13, No. 1, pp 55-70.

GRA-77: Gradwell, F. Why Data Dictionaries? Database. Vol. 6, #2.

GRA-78: Gradwell, F. Objectives and Scope of a Data Dictionary System. Computer Bulletin December 1978.

GRI-75: Grindley, K. Systematics: A New Approach to Systems Analysis. McGraw-Hill Book Co. 1975.

HAM-75: Hammer, M.M., Mcleod, D.J. Semantic Integrity in a Relational Data Base. In: Proceedings of the International Conference on Very Large Data Bases. ACM., September 1975, Vol. 1, #1, pp 25-47.

HAE-78: Haerder, J. Implementing a Generalized Access Path Structure for a Relational Database System. AMC Transactions on Database Systems Vol. 3, #3, September 1978, pp 285-298.

HEA-71: Heath, I.J. Unacceptable File Operations in a Relational Data Base. In: Proceedings of 1971 ACM SIGFIDET Workshop "Data Description Access and Control". pp 19-23.

HUB-75: Hubbard, G., Raver, N. Automating Logical File Design. In: Proceedings of the International Conference on Very Large Data Bases Vol. 1, #1, September 1975, pp 227-253.

JON-77: Jones, P.E. Data Base Design Methodology. A Logical Framework. Q.E.D. Information Sciences Inc. 141 Linden Street

KEN-76: Kent, W. New Criteria For the Conceptual Model. In: Systems For Large Data Bases 1976 Edited by: D.C. Lockemann and E.J. Neuhold. North Holland Publishing Co.

KEN-77: Kent, W. Entities and Relationships in Information. In: Architecture and Models in Data Base Management Systems 1977 Edited by: G.M. Nijssen. North Holland Publishing Co.

KIN-78: King, M.C.F. The Use of Relational Theory as a Design Tool. Systems/Stelsels September 1978

KOB-75: Kobayashi, I. Information and Information Processing Structure. Information Systems Vol. 1, 1975, pp 39-49.

LAN-69: Langefors, B. Concepts, Elementary Files and Data Terms IB-ADB 69 #6.

LAN-70: Langefors, B., Samuelson, K. Information and Data in Systems

LAN-73: Langefors, B. Theoretical Analysis of Information Systems. Averbach Publishers Inc. Philadelphia 1973.

LAN-78: Langefors, B. Comments on a Paper by Biller and Neuhold. Information Systems. Vol. 3, 1978, pp 31-34.

LIN-74: Lindgreen, P. Basic Operations on Information as a Basis For Data Base Design In: Information Processing 74 North Holland Publishing Co.

LIU-71: Liu, H. A Data Base System Defined to Support a Comprehensive Information System. In: Proceedings of 1971 ACM SIGFIDET Workshop on "Data Description, Access and Control pp 349-372.

LUC-74: Lucking, J.R. Data Base Languages in Particular DDL, Development at Codasyl In: ACM SIGMOD Workshop on Data Description Access and Control May 1-3 1974.

MAS-70: Maskell, R. Lexicon - An Established Data Dictionary System. Database Journal. Vol. 6, #7, pp 15-21.

MCG-74: McGee, W.C. A Contribution to the Study of Data Equivalence. In: Data Base Management 1974 Edited by J.W. Klimbie and K.L. Koffeman North Holland Publishing Co., pp 123-130.

MCG-74A: McGee, W.C. A Contribution to the Study of Data Equivalence In: Data Base Management 1974 Edited by: J.W. Klimbie and K.L. Koffeman North Holland Publishing Co. 1974 pp 123-148.

MEA-67: Mealy, G. Another Look at Data. In: AFIPS Fall Joint Computer Conference 1967 pp 525-534.

MEH-74: Mehl, J.W., Wang, C.P. A Study of Order Transformations of Hierarchical Structures in IMS Data Bases In: ACM SIGMOD Workshop on "Data Description Access and Control".May 1-3 1974

MIJ-76: Mijares, I., Peebles, R. A Methodology For the Design of Logical Data Base Structures. In: Modelling in Data Base Systems 1976. Edited by: G.M. Nijssen, North Holland Publishing Co.

MIN-76: Minsley, N. Intentional Resolution of Privacy Protection in Data Base Systems. Communications of ACM. March 1976, Vol. 9, #3, pp 48-58

MIN-78: Minker, J. Binary Relations, Matrices and Inference Developments. Information Systems Vol. 3, 1978, pp 37-47.

MIT-75: Mitoma, M.F. Irani, K.B. Automatic Data Base Schema Design and Optimization. In: Proceedings of the International Conference on Very Large Data Bases Vol. 1, #1, 1975.

MOU-76: Moulin, P. Random J. et al. Conceptual Model as a Data Base Design Tool In: Modelling in Data Base Management Systems. 1976 Edited by : G.M. Nijssen. North Holland Publishing Co.

NIJ-74: Nijssen, G.M. Data Structuring in the DDL and Relational Model In: Data Base Management 1974 Edited by: J.W. Klimbie and K.I. Koffeman. North Holland Publishing Co.

NIJ-75: Nijssen, G.M. Two Major Flaws in the CODASYL DDL 1973 and Proposed Corrections. Information Systems. Vol. 1, 1975,pp 115-132.

NIJ-77: Nijssen, G.M. Current Issued in Conceptual Schema Concepts. In: Architecture and Data Models in Data Base Management Systems 1977 Edited by G.M. Nijssen. North Holland Publishing Co.

NIJ-78: Nijssen, G.M. A Frame Work for Future Data Dictionary Systems. Computer Bulletin December 1978

PIR-77: Pirotte, A. The Entity-Association Model. An Information-Oriented Data Base Model. In: International Symposium 1977. Edited by: E. Morlet and D. Ribbens

POT-70: Potts, G.W. Natural Language Inquiry to an Open-Ended Data Library In: Spring Joint Computer Conference 1970

RIS-77: Rissanen, J. Independent Components of Relations. ACM Transactions of Data Base Systems Vol. 2, #4, December 1977, pp 317-325.

ROU-75: Roussopoulos, N., Mylopoulos, J. Using Semantic Networks for Data Base Management. Proceedings of International Conference on Very Large Data Bases - ACM. Vol. 1, #1, 1975, pp 144-172.

RUC-76: Ruchti, J. Data Descriptions Embedded in Context. In: Modelling Data Base Management Systems. 1976 Edited by G.M. Nijssen.

SCH-75: Schmid, H.A., Swenson, J.R. On the Semantics of the Relational Data Model In: Proceedings ACM - SIGMOD Conference on ??? May 1975

SCH-77: Schmid, H.A. An Analysis of Some Constructs for Conceptual Models. In: Architecture and Models in Data Base Management Systems. 1977 Edited by G.M. Nijssen. North Holland Publishing Co.

SCH-78: Schelling, G. The Use of IBM's Data Dictionary. Computer Bulletin December 1978.

SEN: Senko, M.E. DIA II Semantic Binaries and ANSI SPAMC. In: Data Base Technology. On Line Conferences Limited Uxbridge England pp 1-20

SEN-75: Senko, M.E. Information Systems: Records, Relations, Sets, Entities and Things. Information Systems. Vol. 1, 1975, pp 3-13.

SEN-76: Senko, M.E. DIAM as a Detailed Example of the ANSI SPARC Architecture. In: Modelling in Data Base Management Systems 1976 Edited by: G.M. Nijssen North Holland Publishing Co. pp 73.

SEN-77: Senko, M.E. Conceptual Schemas, Abstract Data Structures, Enterprize Descriptions. In: International Computing Symposium 1977 Edited by E. Morlet and D. Ribbens. North Holland Publishing Co.

SIB-74: Sibley, E.H. On the Equivalences of Data Based Systems. In: ACM SIGMOD Workshop on Data Models, Data Structures Set vs Relational May 1974

SIM-60: Simon A. The New Science of Management Decisions - New York: Harper and Brothers 1960.

SMI-77: Smith, J.M. Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems.Vol. 2, #2, June 1977, pp 105-133.

STA-74: Stacey, G.M. The Interface between a Data Base and its Host Languages. In: Data Base Management 1974. Edited by J.W. Klimbie and K.L. Koffeman. North Holland Publishing Co. pp 105.

STA-76: Stacey, G.M. The Interface between a Data Base and its Host Languages In: Modelling in Data Base Management Systems 1976 Edited by G.M. Nijssen. North Holland Publishing Co.

STA-77: Stamper, R. Physical Objects, Human Discourse and Formal System. In: Architecture and Models in Data Base Management Systems 1977 Edited by G.M. Nijssen. North Holland Publishing Co.

STO-74: Stonebraker, M. A Functional View of Data Independence. In: ACM SIGMOD Workshop on "Data Description, Access and Control". May 1-3 1974.

SUN-74: Sundgren, B. Conceptual Foundation of the Infological Approach to Data Bases In: Data Base Management 1974 Edited by J.W. Klimbie. North Holland Publishing Co.

THO-75: Thomas, D.R. Data Manager - A Free Standing Data Dictionary System. Data Base Journal. Vol. 6, #5, pp 14-17.

TIT-74: Titman, P.J. An Experimental Data Base System Using Binary Relations. In: Data Base Management 1974 Edited by J.W. Klimbie. North Holland Publishing Co.

TOD-76: Todd, S., Owlett, J., Hall, P. Relations and Entities. In: Modelling in Data Base Management Systems. Edited by G.M. Nijssen.

TSI-75: Tsichritzis, D. A Network Framework For Relation Implementation. Computer Systems Research Group University of Toronto.

UHR-73: Uhrowczik, P.P. Data Dictionary/Directories. IBM Systems Journal. Vol. 12, #4, 1973, pp 332-350.

VAN-77: Vandijck, E. Towards a More Familiar Relational Retrieval Language. Information Systems. Vol. 2, 1977, pp 159-169.

VET-77: Vetter, B. Principles of Data Base Systems. In: Proceedings of the International Computing Symposium 1977. Liege Belgium April 4-7. North Holland Publishing Co. pp 555-580.

WAN-75: Wang, C.P., Wedekind, H.H. Segment Synthesis in Logical Data Base Design. IBM Journal of Research and Development. January 1975, pp 71-77.

APPENDIX 1

The following demonstrates how the information content of a job package which contains information on a job may be described by formal message type descriptions. The terminology is that which is used within a telephone company.

Entity Types

E_1 Planned Job - Cable Account Plant Unit

E_2 Job

E_3 Cable Account

E_4 Planned Job - Account - Craft Work Unit

E_5 Account

E_6 Craft

E_7 Planned Job - Account Material Used or Retired

E_8 Planned Job - Account - Labour Type Labour

E_9 Labour Type

E_{10} Planned Job - Account - Accounts Payable Type Expense

E_{11} Accounts Payable Type

E_{12} Planned Usage of Material on Job by Item

E_{13} Outside Plant Material Item Type

E_{14} Day

E_{15} Location

E_{16} Project

E_{17} CPR Section

E_{18} Exchange

E_{19} Employee

E_{20} Engineer

E_{21} Planned Material Placement Work Activity

E_{22} Work Plan

E_{23} Work Plan Location

E_{24} Outside Plant Material Unit

E_{25} Planned Material Retirement Work Activity

E_{26} Planned Total Job - Account Expenditure

Roles

R_1 which includes

R_3 of approval of

R_5 of release of cost estimate of

R_7 of planned receipt of material on site for

R_9 of planned start of construction for

R_{11} of planned ready for service of

R_{13} of planned cutover of

R_{15} of planned start of removal of equipment on

R_{17} of planned completion of removal of equipment on

R_{19} of planned completion of

R_{21} of planned completion of construction on

R_{23} of construction of

R_{25} authorizing

R_{27} supervising

R_{29} placed by

R_{31} retired by

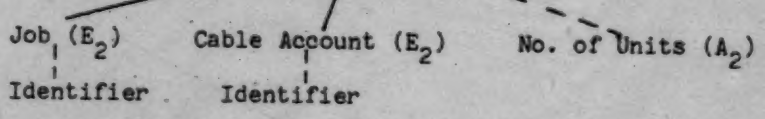
R_{33} originally placing material unit in

Attribute Types

A_1	Identifier
A_2	No. of Units
A_3	Material Length
A_4	Material Placed Value
A_5	Material Retired Value
A_6	Material Salvage Value
A_7	Labour Time
A_8	Value of Expense
A_9	Quantity of Item
A_{10}	Material Amount
A_{11}	Dollar Value of Expense

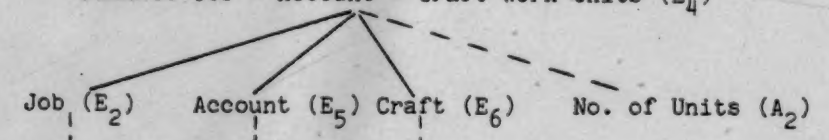
Entity Relationship Types

M_1 Planned Job - Cable Account Plant Unit (E_1)



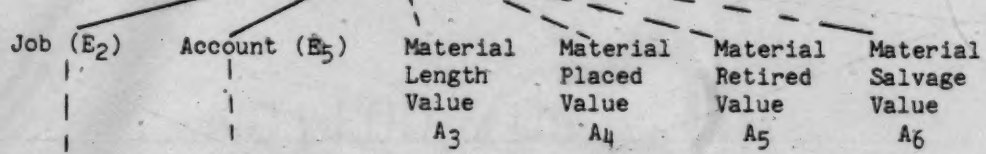
$A_1E_2R_1E_1 \cdot A_1E_3R_1E_1 \cdot A_2E_1$

M_2 Planned Job - Account - Craft Work Units (E_4)



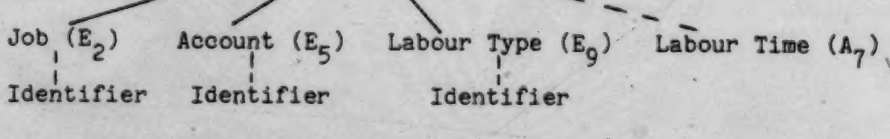
$A_1E_2R_1E_4 \cdot A_1E_5R_1E_4 \cdot A_1E_6R_1E_4 \cdot A_2E_4$

M_3 Planned Job - Account Material Used or Retired (E_7)



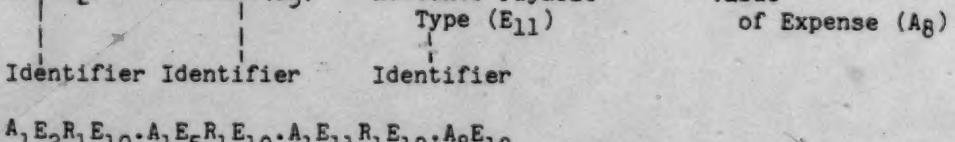
$A_1E_2R_1E_7 \cdot A_1E_5R_1E_7 \cdot A_3E_7 \cdot A_4E_7 \cdot A_5E_7 \cdot A_6E_7$

M_4 Planned Job - Account - Labour Type Labour (E_8)



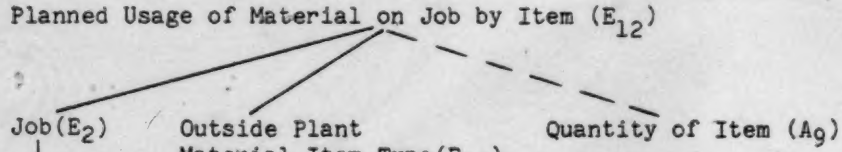
$A_1E_2R_1E_8 \cdot A_1E_5R_1E_8 \cdot A_1E_9R_1E_8 \cdot A_7E_8$

M_5 Planned Job - Account - Accounts Payable Type Expense (E_{10})



$A_1E_2R_1E_{10} \cdot A_1E_5R_1E_{10} \cdot A_1E_{11}R_1E_{10} \cdot A_8E_{10}$

M_6 Planned Usage of Material on Job by Item (E_{12})



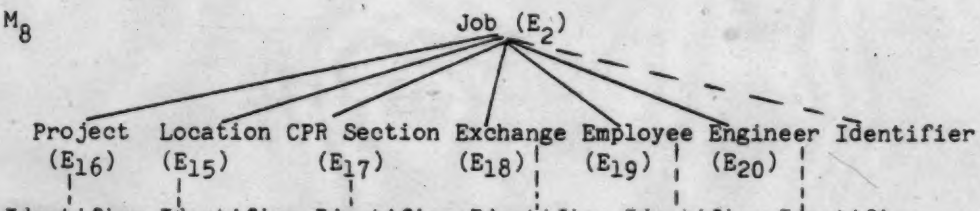
$A_1E_2R_1E_{12} \cdot A_1E_{13}R_1E_{12} \cdot A_9E_{12}$

M_7

$A_1E_{14}R_3E_2 \cdot A_1E_{14}R_5E_2 \cdot A_1E_{14}R_7E_2 \cdot A_1E_{14}R_9E_2 \cdot A_1E_{14}R_{11}E_2$

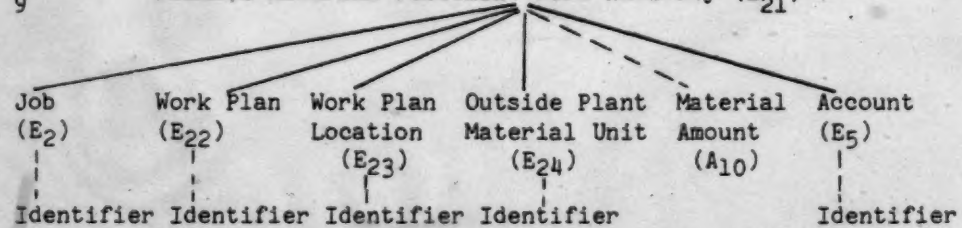
$A_1E_{14}R_{13}E_2 \cdot A_1E_{14}R_{15}E_2 \cdot A_1E_{14}R_{17}E_2 \cdot A_1E_{14}R_{19}E_2 \cdot A_1E_{14}R_{21}E_2 \cdot A_1\ E_2$

M_8

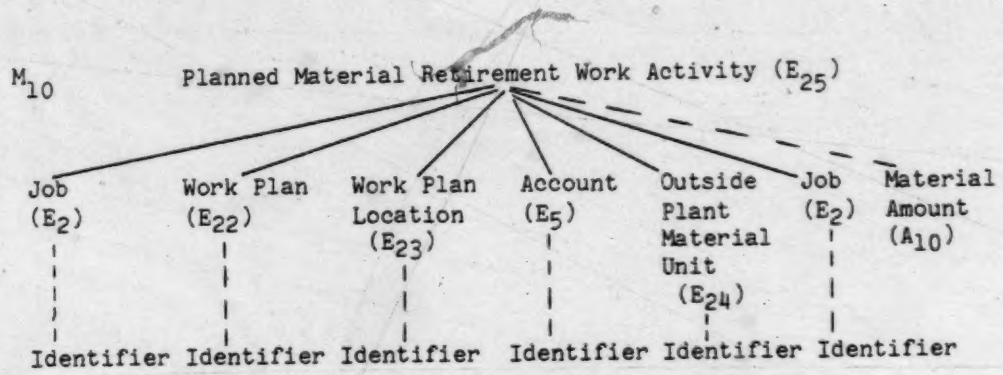


$A_1E_{16}R_1E_2 \cdot A_1E_{15}R_{23}E_2 \cdot A_1E_{17}R_{23}E_2 \cdot A_1E_{18}R_{25}E_2 \cdot A_1E_{19}R_{27}E_2 \cdot A_1E_2$

M_9 Planned Material Placement Work Activity (E_{21})

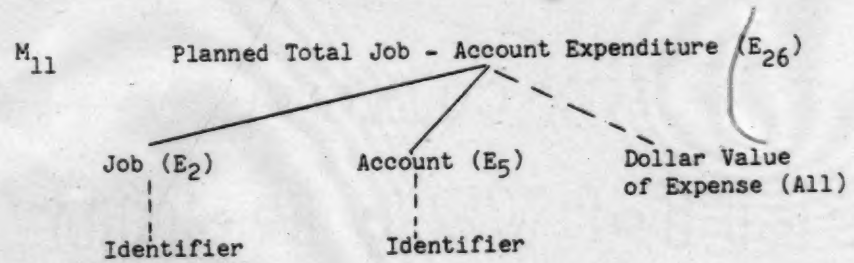


$A_1E_2R_1E_{21} \cdot A_1E_{22}R_1E_{21} \cdot A_1E_{23}R_1E_{21} \cdot A_1E_{24}R_{29}E_{21} \cdot A_{10}E_{21} \cdot A_1E_5R_1E_{21}$



$A_1E_2R_1E_{25} \cdot A_1E_{22}R_1E_{25} \cdot A_1E_{23}R_1E_{25} \cdot A_1E_5R_1E_{25} \cdot A_1E_{24}R_{31}E_{25}$

$A_1E_2R_{33}E_{25} \cdot A_{10}E_{25}$



$A_1E_2R_1E_{26} \cdot A_1E_5R_1E_{26} \cdot A_{11}E_{26}$

The diagrams are not really required and the message type descriptors are

M_1 $A_1E_2R_1E_1 \cdot A_1E_3R_1E_1 \cdot A_2E_1$

M_2 $A_1E_2R_1E_4 \cdot A_1E_5R_1E_4 \cdot A_1E_6R_1E_4 \cdot A_2E_4$

M_3 $A_1E_2R_1E_7 \cdot A_1E_5R_1E_7 \cdot A_3E_7 \cdot A_4E_7 \cdot A_5E_7 \cdot A_6E_7$

M_4 $A_1E_2R_1E_8 \cdot A_1E_5R_1E_8 \cdot A_1E_9R_1E_8 \cdot A_7E_8$

M_5 $A_1E_2R_1E_{10} \cdot A_1E_5R_1E_{10} \cdot A_1E_{11}R_1E_{10} \cdot A_8E_{10}$

M_6 $A_1E_2R_1E_{12} \cdot A_1E_{13}R_1E_{12} \cdot A_9E_{12}$

M_7 $A_1E_{14}R_3E_2 \cdot A_1E_{14}R_5E_2 \cdot A_1E_{14}R_7E_2 \cdot A_1E_{14}R_9E_2 \cdot$
$A_1E_{14}R_{11}E_2 \cdot A_1E_{14}R_{13}E_2 \cdot A_1E_{14}R_{15}E_2 \cdot A_1E_{14}R_{17}E_2 \cdot$
$A_1E_{14}R_{19}E_2 \cdot A_1E_{14}R_{21}E_2 \cdot A_1E_2$

M_8 $A_1E_{16}R_1E_2 \cdot A_1E_{15}R_{23}E_2 \cdot A_1E_{17}R_{23}E_2 \cdot A_1E_{18}R_{25}E_2 \cdot$
$A_1E_{19}R_{27}E_2 \cdot A_1E_2$

M_9 $A_1E_2R_1E_{21} \cdot A_1E_{22}R_1E_{21} \cdot A_1E_{23}R_1E_{23} \cdot A_1E_{24}R_{29}E_{21} \cdot$
$A_{10}E_{21} \cdot A_1E_5R_1E_{21}$

M_{10} $A_1E_2R_1E_{25} \cdot A_1E_{22}R_1E_{25} \cdot A_1E_{23}R_1E_{25} \cdot A_1E_5R_1E_{25} \cdot$
$A_1E_{24}R_{31}E_{25} \cdot A_1E_2R_{33}E_{25} \cdot A_{10}E_{25}$

M_{11} $A_1E_2R_1E_{26} \cdot A_1E_5R_1E_{26} \cdot A_{11}E_{26}$

APPENDIX 2

AN EXAMPLE OF CONVERTING MESSAGE TYPE DESCRIPTION INTO HIERARCHICAL DATA BASES

If we now imagine several occurrences of each of the message types in the preceding appendix we can replace each of them by tuple types as follows.

The variables replacing each component are subscripted such that the first subscript denotes the attribute type, the second the entity type, the third the role type and the fourth the second entity type. A 0 for the third and fourth subscripts indicates no role and no entity type respectively.

M_1 $(D_{1,2,1,1}, D_{1,3,1,1}, D_{2,1,0,0})$

M_2 $(D_{1,2,1,4}, D_{1,5,1,4}, D_{1,6,1,4}, D_{2,4,0,0})$

M_3 $(D_{1,2,1,7}, D_{1,5,1,7}, D_{3,7,0,0}, D_{4,7,0,0}, D_{5,7,0,0}, D_{6,7,0,0})$

M_4 $(D_{1,2,1,8}, D_{1,5,1,8}, D_{1,9,1,8}, D_{7,8,0,0})$

M_5 $(D_{1,2,1,10}, D_{1,5,1,10}, D_{1,11,1,10}, D_{8,10,0,0})$

M_6 $(D_{1,2,1,12}, D_{1,13,1,12}, D_{9,12,0,0})$

M_7 $(D_{1,14,3,2}, D_{1,14,5,2}, D_{1,14,7,2}, D_{1,14,9,2}, D_{1,14,11,2}, D_{1,14,13,2},$
$D_{1,14,15,2}, D_{1,14,17,2}, D_{1,14,19,2}, D_{1,14,21,2}, D_{1,2,0,0})$

M_8 $(D_{1,16,1,2}, D_{1,15,23,2}, D_{1,17,23,2}, D_{1,18,25,2}, D_{1,19,27,2}, D_{1,2,0,0})$

M_9 $(D_{1,2,1,21}, D_{1,22,1,21}, D_{1,23,1,23}, D_{1,24,29,21}, D_{10,21,0,0}, D_{1,5,1,21})$

M_{10} $(D_{1,2,1,25}, D_{1,22,1,25}, D_{1,23,1,25}, D_{1,5,1,25}, D_{1,24,31,25},$
$D_{1,2,33,25}, D_{10,25,0,0})$

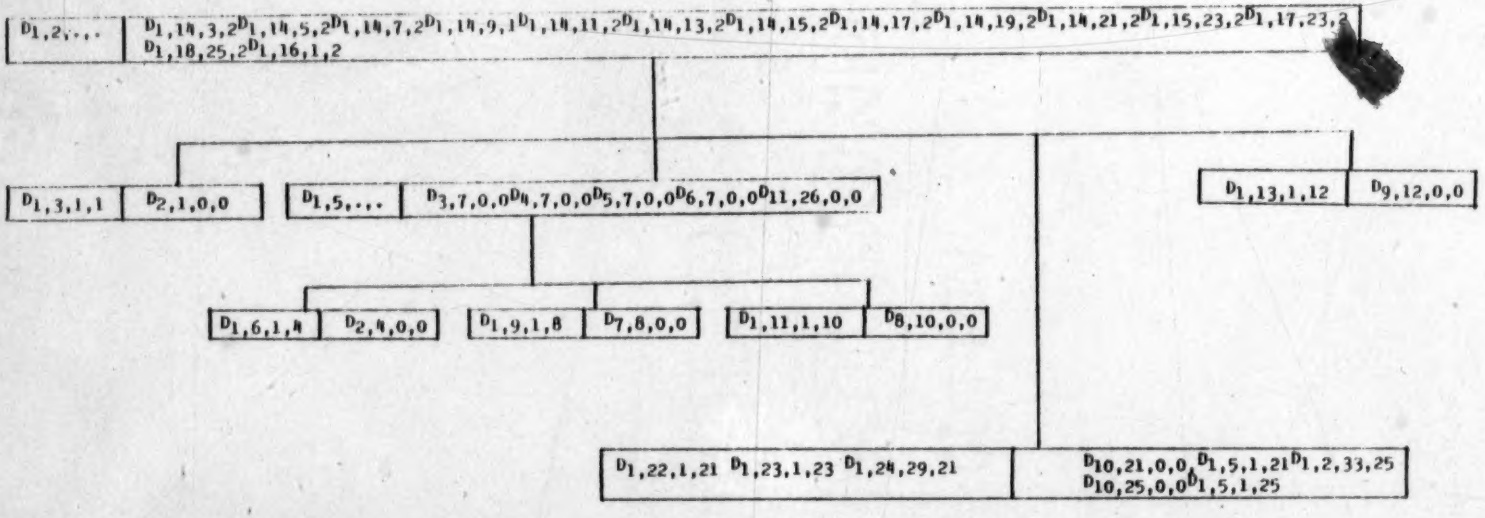
M_{11} $(D_{1,2,1,26}, D_{1,5,1,26}, D_{11,26,0,0})$

The underlined part in each case is the primary key. All relations above except for the relation corresponding to M_8 are in FNF. M_8 is not in FNF because the non-key variables are functionally dependent upon $D_{1,2,0,0}$ only. The primary and only candidate key being the combination consisting of $D_{1,2,0,0}$ and $D_{1,19,27,2}$. Replacing M_8 by two primary relations $M_{8,1}$ and $M_{8,2}$ we get

$M_{8,1}$ ($\underline{D_{1,2,0,0}}$, $D_{1,16,1,2}$, $D_{1,15,23,2}$, $D_{1,17,23,2}$, $D_{1,18,25,2}$)

$M_{8,2}$ ($\underline{D_{1,2,0,0}, D_{1,19,27,2}}$)

Let us now build the sets of tuples into a hierarchical data structure. This structure, consisting of two hierarchical data bases, is shown on the following page.



$D_{1,2,0,0}$ $D_{1,19,27,2}$

* The two's indicates that these subscripts vary depending upon the path.

** The 1 indicates the end of the key.